LA-UR-03-2228

Approved for public release; distribution is unlimited.

Title: MCNP5 PARALLEL PROCESSING WORKSHOP

Author(s): FORREST B. BROWN
J. TIM GOORLEY
JEREMY E. SWEEZY

Submitted to: ANS Mathematics & Computation Topical Meeting, Gatlinburg, TN, April 11,2003

MCNP5 Parallel Processing Workshop

Forrest B. Brown, J. Tim Goorley, Jeremy E. Sweezy
MCNP Monte Carlo Development Team
Diagnostics Applications Group (X-5)
Los Alamos National Laboratory

Abstract

MCNP5 Parallel Processing Workshop

Forrest B. Brown, J. Tim Goorley, Jeremy E. Sweezy
MCNP Monte Carlo Development Team
Diagnostics Applications Group (X-5)
Los Alamos National Laboratory

American Nuclear Society
Mathematics & Computation Topical Meeting
Gatlinburg, TN, April 11, 2003

After a brief review of parallel processing, the specific uses of message-passing and threading in MCNP5 will be discussed. Practical topics to be covered include system configuration, MCNP5 compilation, MCNP5 running strategies for mixed parallelism (MPI+threads), MCNP5 on PC clusters, MCNP5 on Linux clusters, MCNP5 on large parallel systems, MPI for PC clusters & larger systems, OpenMP threading, PVM, and Fortran-90. Examples will be demonstrated on a PC (laptop) cluster.

Part I – Parallel Processing & Monte Carlo (Brown)
- Parallel Computing
- Monte Carlo
- MCNP5 Parallelism
- MCNP5 on Tera-scale ASCI Systems

Part II – MCNP5 & PC Clusters – Windows (Goorley)
- MCNP5, MPI, & PVM
- Windows clusters
- Demo

Part III – MCNP5 & PC Clusters – Linux (Sweezy)
- MCNP5, MPI, & PVM
- Linux clusters
- Demo

Part I – Parallel Processing & Monte Carlo

Forrest B. Brown

Part I – Parallel Processing & Monte Carlo

- MCNP5 Overview
- Parallel Computing
  - Parallel Computers
  - Message Passing
  - Threads
  - Amdahl's Law
- Parallel Monte Carlo
  - Parallel Algorithms
  - Histories, Random Numbers, Tallies
  - Load Balancing, Fault Tolerance, …
  - Parallel Performance & Scaling
- MCNP5 Parallel Processing
  - MCNP5 parallelism
  - MPI or PVM + Threads
  - Run Commands & Input Options
  - Performance on ASCI Tera-scale systems

mcnp 5

MCNP Tradition at Los Alamos

- The **MCNP** Monte Carlo radiation transport code has been developed and supported by the Monte Carlo team at LANL for **25 years**.
- Concurrently, the extensive nuclear and atomic **data libraries** have also been under constant development
- This tradition continues in the **Eolus ASCI Project** and related efforts in the Diagnostics Applications Group (X-5)
  - 12 MCNP code developers
  - Physical Data team also in X-5
  - Two application teams (user groups) in X-5

MCNP Development Team

Monte Carlo Development

Forrest Brown (Team Lead)

Tom Booth	Jeffrey Bull	Larry Cox
Art Forster	Tim Goorley	Grady Hughes
Russell Mosteller	Richard Prael	Elizabeth Selcow
Avneet Sood	Jeremy Sweezy	

Computer Support

Susan Post	Richard Barrett	Brian Jean
Teri Roberts	Skip Egdorf	Mark Zander

Research Associates

Taro Ueki (postdoc)

X-5 Data Team

Robert Little	Stephanie Frankle	Morgan White
Joanne Campbell	Stepan Mashnik	

University R&D	William Martin	Jerry Spanier
High-Energy Physics	Nikolai Mokhov	Sergei Strepanov
Visual Editor	Randy Schwarz	Lee Carter

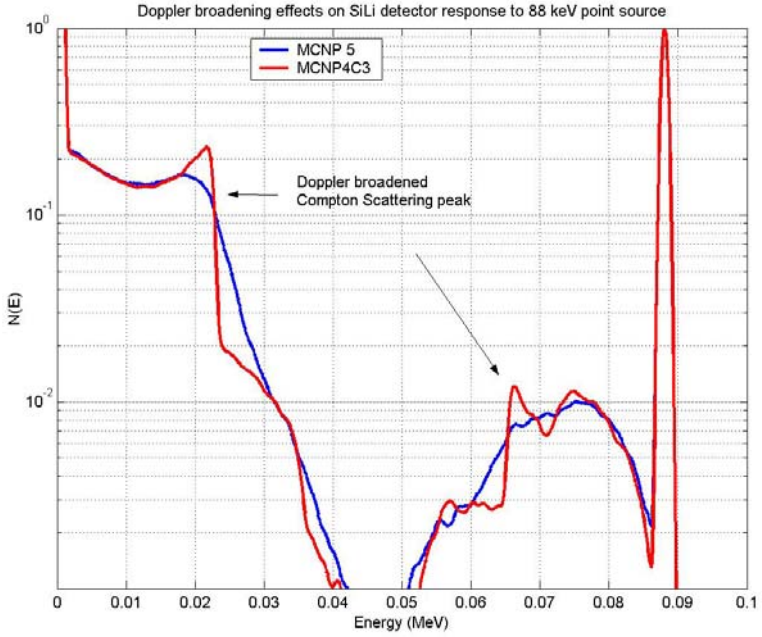
MCNP Version 5

- Modernization of MCNP: 2-year effort driven by ASCI Program needs for
  - **Modern software engineering (SE) & software quality assurance (SQA),**
  - **Strict adherence to standards for Fortran-90 & parallel processing,**
  - **Preservation of all existing code capabilities,**
  - **Flexibility for rapid introduction of new features and advanced computers.**
  - An evolutionary approach to MCNP modernization was followed, to minimize the chances of introducing new errors.
- MCNP5 is a rewrite of MCNP4C
  - Entire code is standard Fortran-90
  - Standard parallel coding: MPI (message-passing) + OpenMP (threads)
  - Fortran-90 dynamic memory allocation
  - Vastly improved modern coding style: spaces, blank lines, modules, replaced many thousands of GOTO's, ...
  - Some new features & new physics

New Features in MCNP Version 5

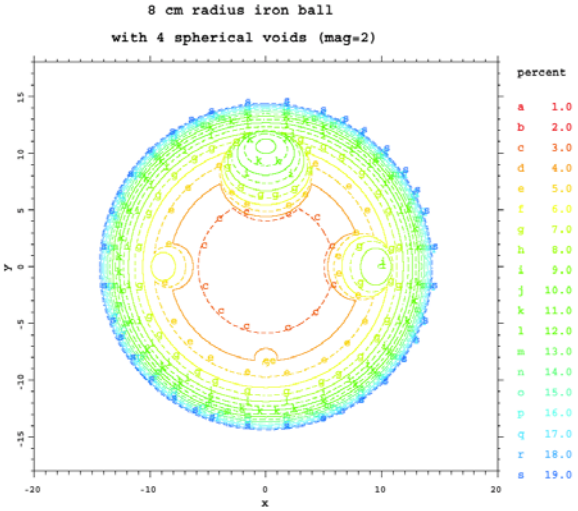
- Doppler Energy Broadening for Photon Transport
- Mesh Tallies
- Neutral Particle Image Tallies
- Sources: translate/rotate/repeat, Gaussian, particle type
- Easier specification of sources in repeated structures
- Time & energy splitting/rouletting
- Enhanced Parallel Processing Support
- Extended Random Number Package
- Unix-based build system, using GNU make
- Pulse height tally variance reduction (Spring, 2003)
- Radioisotope sources (Spring, 2003)
- Improved plotting options & more colors

Doppler Energy Broadening for Photons



Neutral Particle Image Tallies

- Release of long-term LANL feature
- Neutron and Photon radiography uses a raster of point detectors
- Each source and collision event contributes to all points
- Radiography Image Cards
  - FIR – flux image radiograph
  - FIP – flux image pinhole
  - FIC – flux image cylinder
- Two plotting options
  - Color contours
  - Color filled

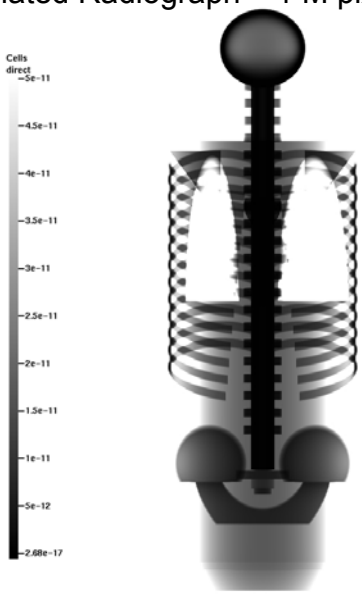


An unscattered image from a 1 million pixel tally with a 6 MeV photon point source

Neutral Particle Image Tallies

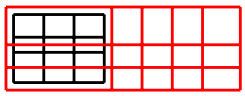
MCNP Model of Human Torso

Simulated Radiograph – 1 M pixels



Tally & Criticality Safety Enhancements

- Mesh Tallies
  - Arbitrary XYZ or RZ mesh superimposed on problem geometry
  - Multiple mesh tallies permitted, with separate mesh for each
  - Easy way to get assembly-tallies, dose fields, images, etc.



- Criticality Safety Parameters
  - Average energy of neutrons causing fission
  - Energy corresponding to average lethargy of neutrons causing fission
  - Fission to absorption ratios, etc.

New Random Number Generator

- Traditional MCNP RN generator
  - Based on 48-bit integers
  - Period ~10^{14} , stride = 152917
  - RNs reused after 500M histories
- New RN Generator
  - Based on 63-bit integers
  - Completely portable Fortran-90, modular
  - Efficient skip-ahead algorithm
  - Period ~10^{19} – 10,000x longer than previous
- Tested extensively
  - Knuth's statistical tests
  - Marsaglia's DIEHARD tests
  - Spectral test
- For now, traditional RN generator is default, new RN generators optional (will change next year)

Computer Systems Supported

- SGI IRIX64
- IBM AIX
- HP/Compaq OSF1
- Sun SunOS
- Linux with Absoft compiler
- Linux with Lahey compiler
- Linux with Portland Group compiler
- Windows PC with CVF compiler
- Windows PC with Absoft compiler
- Windows PC with Lahey compiler
- X11 graphics – all systems
- Mac OSX with Absoft compiler (soon)
- Itanium with Intel compiler (probably)

Must have Fortran-90 compiler – not F77 or g77

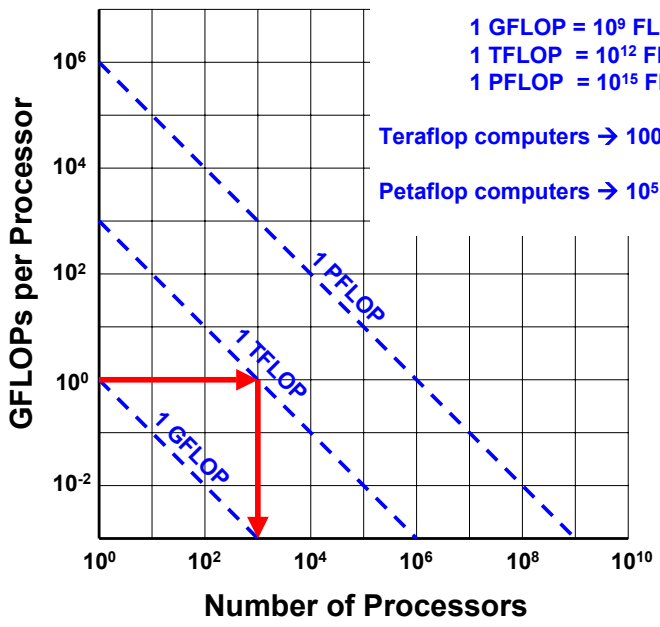
Parallel Computing

Trends in Computing Technology

- Commodity chips:
  - Microprocessor speed → ~2x gain / 18 months
  - Memory size → ~2x gain / 18 months
  - Memory latency → ~ no change (getting worse)
- High-end scientific computing
  - Key driver (or limit) → **economics:** mass production of desktop PCs & commercial servers
  - Architecture → **clusters**: with small/moderate number of commodity microprocessors on each node
- Operating systems
  - Desktop & server → Windows, Linux
  - Supercomputers → Unix, Linux

CPU performance on supercomputer ➔ same as desktop PC
High-performance scientific computing ➔ parallel computing

Parallel Computers



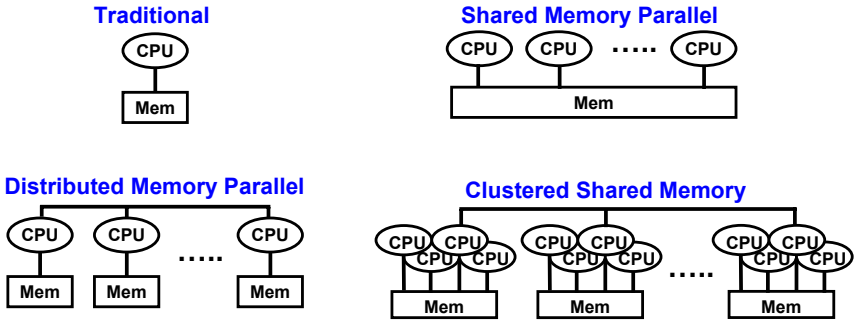
1 GFLOP = 10^9 FLOP
1 TFLOP = 10^{12} FLOP
1 PFLOP = 10^{15} FLOP

Teraflop computers → 1000's of processors

Petaflop computers → 10^5 processors ?

Parallel Computers

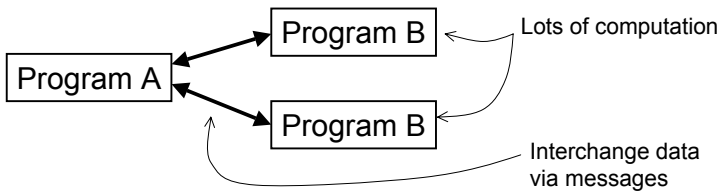
- Characterize computers by:
  - CPU: scalar, vector, superscalar, RISC, ...
  - Memory: shared, distributed, cache, banks, bandwidth, ...
  - Interconnects: bus, switch, ring, grid, ...
- Basic types:



Approaches to Parallel Processing

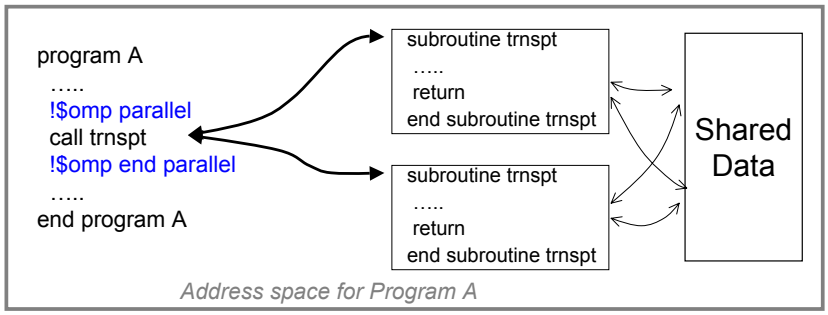
High-level	• Independent programs + **message-passing** • Distribute work among processors • Loosely-coupled • Programmer must modify high-level algorithms
Mid-level	• **Threads** (task-level) • Independent tasks (subprograms) + **shared memory** • For shared memory access, use locks on critical regions • Compiler directives by programmers
Low-level	• **Threads** (loop-level) • Split DO-loop into pieces, compute, synchronize • Compiler directives by programmers
Low-level	• Pipelining or vectorization • Pipelined execution of DO-loops • Automatic vectorization by compilers &/or hardware, or compiler directives by programmers

Message-passing



- Independent programs
- Separate memory address space for each program (private memory)
- All control information & data must be passed between programs by explicit messages (SENDs & RECEIVEs)
- Can run on distributed or shared memory systems
- Efficient only when $T_{computation} >> T_{messages}$
- Standard message-passing:
  - MPI
  - PVM

Threading (task-level)



- Single program, independent sections or subprograms
- Each thread executes a portion of the program
- Common address space, must distinguish private & shared data
- Critical sections must be "locked"
- Can run only on shared memory systems, not distributed memory
- Thread control by means of compiler directives
- Standard threading:
  - OpenMP

Threading (loop-level)

```
!omp do parallel
do k=1,n
  c(k) = a(k)+b(k)
enddo
```

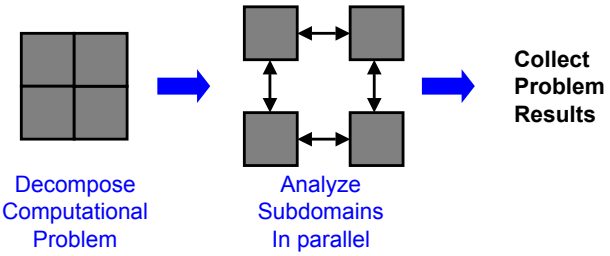
→

```
do k=1,n,2
  c(k) = a(k)+b(k)
enddo
```

```
do k=2,n,2
  c(k) = a(k)+b(k)
enddo
```

- Single DO-loop within program
- Each loop iteration must be independent
- Each thread executes different portion of DO-loop
- Invoked via compiler directives
- Standard threading:
  - OpenMP

Domain Decomposition



- Coarse-grained parallelism, high-level
- For mesh-based programs:
  1. Partition physical problem into blocks (domains)
  2. Solve blocks separately (in parallel)
  3. Exchange boundary values as needed
  4. Iterate on global solution
- Revised iteration scheme may affect convergence rates
- Domain decomposition is often used when the entire problem will not fit in the memory of a single SMP node

Amdahl's Law

If a computation has fast (parallel) and slow (scalar) components, the overall calculation time will be dominated by the slower component

$$\text{Overall System Performance} = \text{Single CPU Performance} * \frac{1}{1-F + F/N}$$

where **F** = fraction of work performed in parallel
N = number of parallel processors

Speedup = 1 / (1-F + F/N)

For N=10

F	S	F	S
20%	1.2	90%	5.3
40%	1.6	95%	6.9
60%	2.2	99%	9.2
80%	3.6	99.5%	9.6

For N=infinity

F	S	F	S
20%	1.3	90%	10
40%	1.7	95%	20
60%	2.5	99%	100
80%	5	99.5%	200

Amdahl's Law

My favorite example …..

Which system is faster?

System A: (16 processors)•(1 GFLOP each) = **16 GFLOP total**

System B: (10,000 procs)•(100 MFLOP each) = **1,000 GFLOP total**

Apply Amdahl's law, solve for F:

$$1 / (1-F + F/16) = .1 / (1-F + F/10000)$$

➔ **System A is faster, unless >99.3% of work is parallel**

- In general, a smaller number of fatter nodes is better
- For effective parallel speedups, must parallelize everything

Parallel Monte Carlo

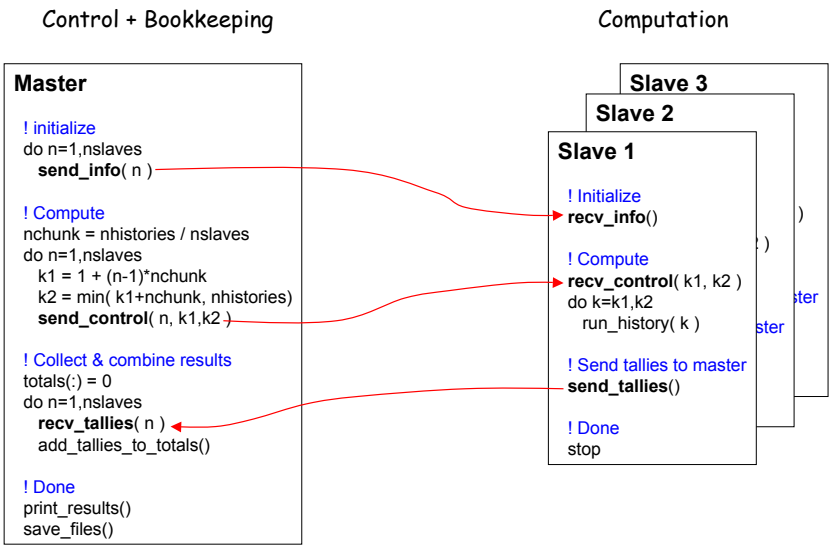
Parallel Algorithms

- Possible parallel schemes:
  - Jobs — run many sequential MC calculations, combine results
  - Functional — sources, tallies, geometry, collisions, ...
  - Phase space — space, angle, energy
  - Histories — Divide total number of histories among processors
- All successful parallel Monte Carlo algorithms to date have been history-based.
  - Parallel jobs always works, variation on parallel histories
  - Some limited success with spatial domain decomposition

Master / Slave Algorithm (Simple)

- Master task: control + combine tallies from each slave
- Slave tasks: Run histories, tallies in private memory
  - **Initialize**:
    Master sends problem description to each slave
    (geometry, tally specs, material definitions, …)
  - **Compute**, on each of N slaves:
    Each slave task runs 1/N of total histories.
    Tallies in private memory.
    Send tally results back to Master.
  - **Combine tallies:**
    Master receives tallies from each slave &
    combines them into overall results.
- Concerns:
  - Random number usage
  - Load-balancing
  - Fault tolerance (rendezvous for checkpoint)
  - Scaling

Master / Slave Algorithm (Simple)



Random Number Usage

- Linear Congruential RN Generator

  $S_{k+1} = g\, S_k + C \mod 2^M$

- RN Sequence & Particle Histories

  1 2 3 etc.

  MCNP stride for new history: 152,917

- To skip ahead k steps in the RN sequence:

  $S_k = g\, S_{k-1} + C \mod 2^M = g^k S_0 + C\,(g^k-1)/(g-1) \mod 2^M$

- Initial seed for n-th history

  $S_0^{(n)} = g^{n*152917} S_0 + C\,(g^{n*152917}-1)/(g-1) \mod 2^M$

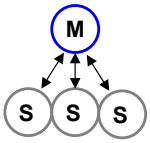
  This is easy to compute quickly using exact integer arithmetic

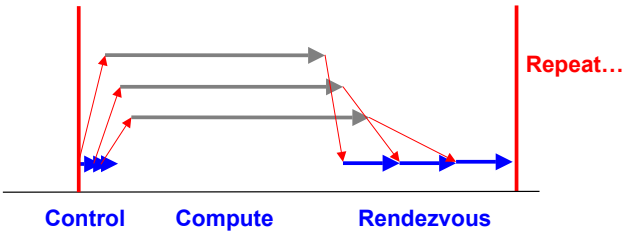
- Each history has a unique number
  - Initial problem seed → initial seed for n^{th} particle on m^{th} processor
  - If slave knows initial problem seed & unique history number, can initialize RN generator for that history

Fault Tolerance

- On parallel systems with complex system software & many CPUs, interconnects, disks, memory, MTBF for system is a major concern.
- Simplest approach to fault tolerance:
  - Dump checkpoint files every M histories (or XX minutes)
  - If system crashes, restart problem from last checkpoint
- Algorithm considerations
  - Rendezvous every M histories.
  - Slaves send current state to master, master saves checkpoint files
  - Parallel efficiency affected by M.

Fault Tolerance





- For efficiency, want (compute time) >> (rendezvous time)
  - Compute time: Proportional to #histories/task
  - Rendezvous time: Depends on amount of tally data & latency+bandwidth for message-passing

Master / Slave Algorithm, with Rendezvous

- **Initialize**:
  Master sends problem description to each slave
  (geometry, tally specs, material definitions, …)

- For rendezvous = 1, L

  - **Compute**, on each of N slaves:
    Each slave task runs 1/N of (total histories)/L.
    Tallies in private memory.
    Send tally results back to Master.

  - **Combine tallies:**
    Master receives tallies from each slave &
    combines them into overall results.
  - **Checkpoint:**
    Master saves current tallies & restart info in file(s)

Load Balancing

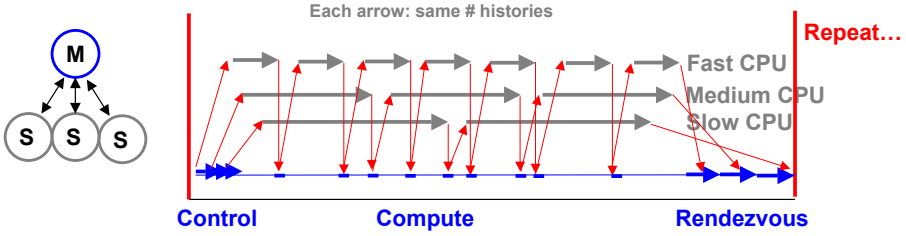
- Time per history may vary significantly
  - For problems using variance reduction:
    - Particles headed in "wrong" direction may be killed quickly, leading to a short history.
    - Particles headed in "right" direction may be split repeatedly. Since the split particles created are part of the same history, may give a very long history.
  - For problems run on a workstation cluster:
    - Workstation nodes in the cluster may have different CPU speeds
    - Workstations in the cluster may be simultaneously used for interactive work, with highly variable CPU usage on that node.
    - Node performance effectively varies continuously over time.
- Naïve solution
  - Monitor performance per node (e.g., histories/minute)
  - Periodically adjust number of histories assigned to each node, according to node performance

  # histories assigned to node n ~ measured speed of node n
- Better solution: self-scheduling

Load Balancing – Self-Scheduling

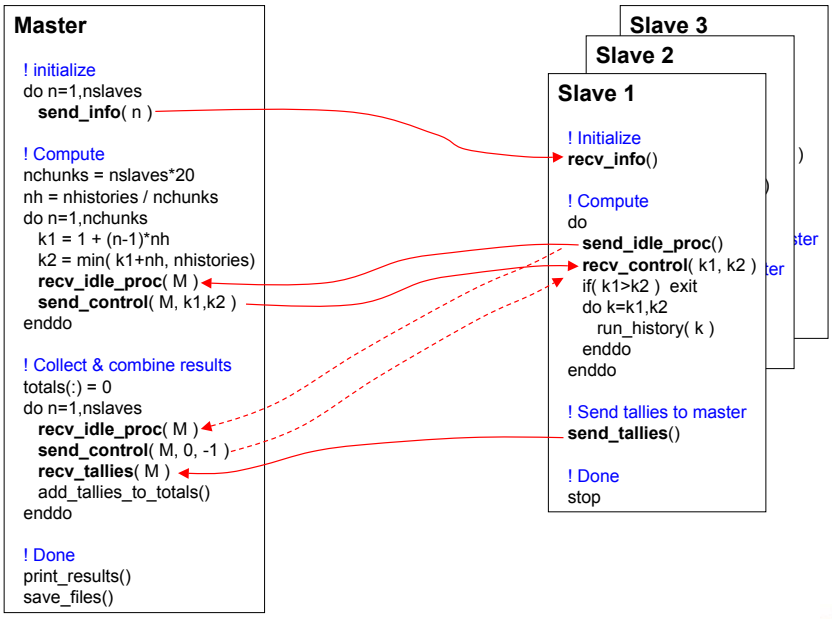
- For a problem with N slave processors, divide histories into **more than** N chunks.
  - Let L = number of chunks, L > N
  - Typically, L ~ 20 N or L ~ 30 N
  - Histories/chunk = (total histories) / L
  - Slave: If idle, ask master for work. Repeat until no more work.
  - Master: Send chunk of work to idle slave. Repeat until no more work.
  - On average, imbalance in workload should be < 1/L
- Additional gains:
  - Naïve master/slave algorithm is **synchronous**
  - Self-scheduling master/slave algorithm is **asynchronous**. More overlap of communication & computation ➔ reduced wait times & better performance

Load Balancing – Self-Scheduling



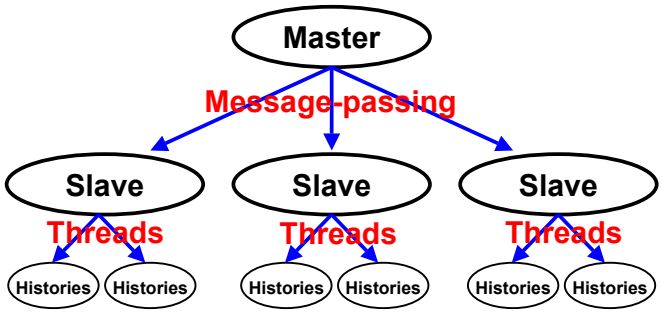
- Much more communication with Master, but only minimal amount of control info needed (1st & last history in chunk)
- Need to handle stopping condition carefully – avoid "dangling" messages

Load Balancing – Self-Scheduling



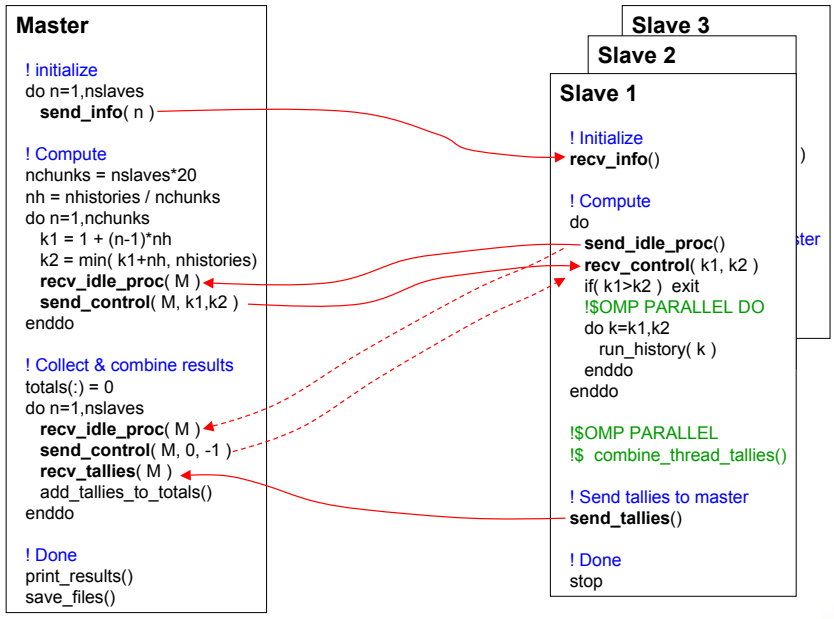
Hierarchical Parallelism

- For clustered SMPs,
  - Use message-passing to distribute work among slaves ("boxes")
  - Use threading to distribute histories among individual processors on box



- Only the master thread (thread 0) on each slave uses MPI send/recv's

Master / Slave Algorithm, threaded & self-scheduling



Parallel Monte Carlo Performance

Parallel MC Computational Characteristics

- For master/slave algorithms (with self-scheduling, fault tolerance, & threads):
  - **No** communication among slave tasks
  - Occasional communication between master & slaves (rendezvous)
  - Slave tasks are compute-intensive
    - Few DO-loops
    - 40% of ops are test+branch (IF… GOTO…)
    - Irregular memory access, no repetitive patterns
  - For fixed-source problems:
    - Only 1 rendezvous is strictly necessary, at end of calculation
    - More rendezvous used in practice, for fault tolerance
  - For eigenvalue problems (K-effective):
    - Must have a rendezvous every cycle (cycle = batch = generation)
    - Master controls iteration & source sampling
- Common-sense approach to performance:
  Fewer rendezvous ➔ better parallel performance

Parallel MC Performance Measures

- Metrics
  - Speedup $S_N = T_1 / T_N$ N = # processors
  - Efficiency $E_N = S_N / N$
- Fixed overall work (fixed problem size)
  - Efficiency decreases with N
  - Speedup (eventually) drops as N increases
  - Why?
    As N increases, same communication/processor, but less work/processor (fewer histories/processor) ➔ (computation/communication) decreases
- Fixed work per processor (scaled problem size)
  - Efficiency approx. constant with N
  - Speedup approx. linear with N
  - Why?
    As N increases, same communication/processor, same work/processor (# histories ~ N) ➔ (computation/communication) stays approx. same
  - Called **scaled speedup**

Parallel MC Performance Limits

- Another way to determine effiency

Parallel Efficiency = $T_C / (T_C + T_M)$

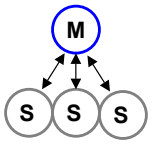
T_C = computing time
T_M = time for messages, not overlapped with computing

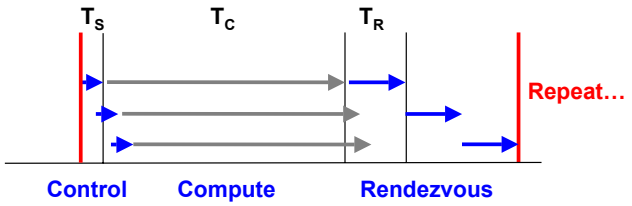
- Slaves can send messages in parallel

- Master receives & processes messages serially

If enough messages are sent to master, extra wait time will limit performance

Parallel MC Performance Scaling





N = # processors
T_1 = CPU time for M histories using 1 processor
(Depends on physics, geometry, compiler, CPU speed, memory, etc.)
L = amount of data sent from 1 slave each rendezvous

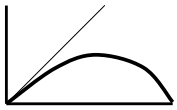
$T_S = 0$ negligible, time to distribute control info

$T_R = s + L/r$ **s** = latency for message, **r** = streaming rate

$T_C^{fix} = T_1 / N$ fixed problem size, **M** histories/rendezvous
$T_C^{scale} = T_1$ scaled problem size, **NM** histories/rendezvous

Parallel MC Performance Scaling

- Scaling models, for master/slave with serial rendezvous
  - "fixed" = constant number of histories/rendezvous, M (constant work)
  - "scaled" = M histories/slave per rendezvous, NM total (constant time)

Hist./rendezvous	Speedup	
fixed	$S = N / (1 + cN^2)$	
scaled	$S = N / (1 + cN)$	

N = number of slaves
$c = (s + L/r) / T_1$

$T_1 \sim M$, more histories/rendezvous ➔ larger T_1, smaller c
S+L/r, fixed, determined by number of tallies, ….

As M→infinity, c→0, S→N (limit for 1 rendezvous)

Parallel MC Summary

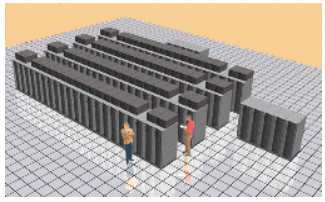
- Master/slave algorithms work well
  - Load-balancing: Self-scheduling
  - Fault-tolerance: Periodic rendezvous
  - Random numbers: Easy, with LCG & fast skip-ahead algorithm
  - Tallies: Use OpenMP "critical sections"
  - Scaling: Simple model, more histories/slave + fewer rendezvous
  - Hierarchical: Master/slave MPI, OpenMP threaded slaves
  - Portability: MPI/OpenMP, clusters of anything

- Remaining difficulties
  - Memory size: Entire problem must fit on each slave
    - Domain-decomposition has had limited success
      - Should be OK for reactor problems
      - May not scale well for shielding or time-dependent problems
      - For general 3D geometry, effective domain-decomposition is unsolved problem
    - Random access to memory distributed across nodes gives huge slowdown
      - May need functional parallelism with "data servers"

MCNP5 Parallel Calculations

Advanced Simulation & Computing Initiative – ASCI

Blue Pacific – 3 TeraOps



Red – 3 TeraOps

Blue Mountain – 3 TeraOps

White – 12 TeraOps

Q – 20 TeraOps

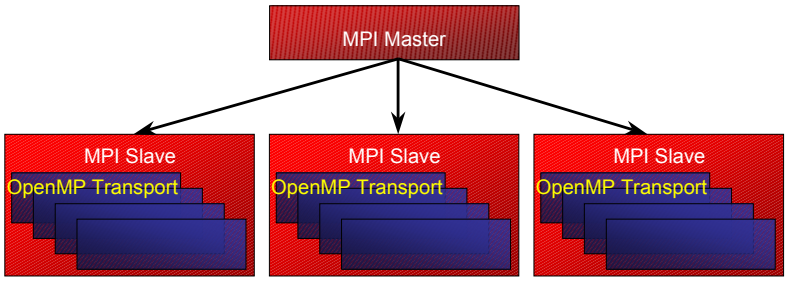
Parallelism in MCNP5

- Distributed parallelism (message passing): MPI or PVM
- Shared-memory parallelism (threads): OpenMP
- Mixed parallelism supported: MPI/OpenMP, PVM/OpenMP

- Support for all modes on PC, Unix, & Linux

- MPI/OpenMP combination used regularly at LANL on 1000+ procs

- Answers obtained in parallel are expected to, and mostly do, track sequential calculations exactly



Parallelism in MCNP5

- We routinely test MCNP 5 with MPI/OpenMP on:
  - **ASCI Bluemountain** – SGI IRIX64, 48 boxes x 128 processors/box
    - 1,000 processor jobs are "routine"
  - **ASCI White** – IBM AIX, 512 boxes x 16 processors/box
  - **ASCI Q** – HP/Compaq OSF1, 2 x 512 boxes x 4 processors/box
  - **Linux cluster**
  - **Windows PC cluster**

Parallelism in MCNP5

- Threading
  - Individual histories are handelled by separate threads
  - No thread syncrhonization is needed during a history
  - Implemented by OpenMP compiler directives
  - Tallies, RN data, & some temporary variables for history are in thread-private memory

    Example:

```
common /RN_THREAD/ RN_SEED, RN_COUNT, RN_NPS
!$OMP THREADPRIVATE ( /RN_THREAD/ )
save                          /RN_THREAD/
```

  - OpenMp **critical sections** are used for some tallies or variable updates

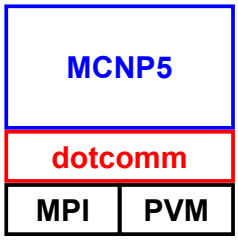
    Example:

```
!$OMP CRITICAL (RN_STATS)
RN_COUNT_TOTAL = RN_COUNT_TOTAL + RN_COUNT
$!OMP END CRITICAL (RN_STATS)
```

  - Message-passing & file I/O are executed only from thread-0 (master thread) for each MPI task

Parallelism in MCNP5

- Message-passing
  - In MCNP5, all message-passing is handled by calls to the **dotcomm** package, a communications layer which contains an interface to either MPI or PVM



  - Either MPI or PVM message-passing is selected in **dotcomm** at compile-time
  - Using the **dotcomm** package & either MPI or PVM, MCNP5 can run in parallel without source code changes on
    - Parallel supercomputers (e.g., ASCI tera-scale computers)
    - COWs (clusters of workstations)
    - Linux clusters
    - PC clusters

Parallelism in MCNP5

- PVM Message-passing using **dotcomm**
  - We have had difficulties with PVM performance on some systems due to the way PVM allocates communications buffers dynamically.
  - To solve these problems, all dynamic storage allocation for communicaitons buffers is handled by **dotcomm**. Messages are buffered & constructed within **dotcomm**, & then PVM is used to send them.
  - This design choice significantly improves performance on some systems, but bypasses PVM's native ability to convert data to "network standard" when running on a heterogeneous cluster.
  - As a result, MCNP5 is restricted to clusters where all machines have the same native data types, e.g., all machines are big-endian IEEE or all machines are little-endian IEEE. Machines in a cluster can be from different vendors (e.g., IBM, Sun).
- MPI & performance
  - MPI is a standard for message-passing parallelism
  - Many parallel computer vendors have optimized MPI for their systems, to take advantage of unique hardware characteristics in the interconnect systems & significantly improve message-passing latency/bandwidth.

MCNP5 Parallel Calculations

N = total number of MPI tasks, master + (N-1) slaves

M = number of OpenMP threads/slave

- Running on parallel systems with MPI only

  mpirun -np N mcnp5.mpi i=inp01 .....

- Running with threads only

  mcnp5 tasks M i=inp01 .....

- Running on parallel systems with MPI & threads

  ASCI Bluemountain (SGI)

  mpirun -np N mcnp5.mpi tasks M i=inp01 .....

  ASCI Q (HP/Compaq)

  prun –n N -c M mcnp5.mpi tasks M i=...

  If submitting jobs through a batch system (e.g., LSF), N & M must be consistent with LSF requested resources

MCNP5 Parallel Calculations

- MPI or PVM ?
  - MPI is a standard
  - Many vendors optimize MPI performance for their systems
  - MPI is available for all machines (ASCI, parallel, clusters, Linux, PCs,…)
  - ASCI Program requires MPI
  - PVM has a long, successful history
  - PVM is the only choice for heterogeneous clusters of big-endian & little-endian machines
    [But, MCNP5 doesn't currently support such big/little endian mixes]
  - We support both MPI & PVM
  - We recommend MPI – that's what we routinely test & use at LANL

MCNP5 Parallel Calculations

- How many threads ?
  - Max number of threads = # CPUs per node
    - ASCI Bluemountain: 128 cpus / node
    - ASCI Q: 4 cpus /node
    - Laptop PC cluster: 1 cpu / node
  - Experience on many systems has shown that a moderate number of threads per slave is efficient; using too many degrades performance
    - ASCI Bluemountain: 4-12 threads/slave usually effective
      >16 threads/slave usually has bad performance
    - ASCI Q: 4 threads/slave is effective
  - Rules-of-thumb vary for each system
    - Thread efficiency is strongly affected by operating system design
    - Scheduling algorithm for threads used by operating system is generally designed to be efficient for small number of threads (<16)
    - For large number of threads, context-switching & cache management may take excessive time, giving poor performance
    - Other jobs on system (& their priority) affect thread performance
    - No definite rules – need to experiment with different numbers of threads

MCNP5 Parallel Calculations

- Parallel performance is sensitive to number of rendezvous
  - Can't control number of rendezvous directly
  - The following things cause a rendezvous:
    - Printing tallies
    - Dumping to the RUNTPE file
    - Tally Fluctuation Chart (TFC) entries
    - Each cycle of eigenvalue problem

- Use PRDMP card to minimize print/dump/TFC

PRDMP ndp ndm mct ndmp dmmp

ndp = increment for printing tallies ← **use large number**
ndm = increment for dump to RUNTPE ← **use large number**
mct = flag to suppress time/date info in MCTAL
ndmp = max number of dumps in RUNTPE
dmmp = increment for TFC & rendezvous ← **use large number**

For fixed-source problems, increments are in particles
For eigenvalue problems, increments are in cycles

MCNP5 Parallel Calculations

- Keff calculations: Use KCODE card for hist/cycle

  - Want to reduce the number of cycles
  - More histories in each cycle
  - Should run hundreds of cycles or more for good results

KCODE nsrck rkk ikz kct …..

nsrck = histories / cycle ← **use a large number**
rkk = initial guess for Keff
ikz = number of initital cycles to discard
kct = total number of cycles to run

Suggested: nsrck ~ (thousands) x (number of processors)

MCNP5 Parallel Calculations

- Running large, parallel jobs on ASCI systems

1,000+ processor job on ASCI Bluemountain
- 48 boxes x 128 cpu/box
- LSF limits jobs to 126 cpu/box

- Each box: 12 MPI tasks x 10 threads/MPItask = 120 cpus/box

- Overall: 126 cpu/box x 9 boxes = 1134 cpus
  12 x 9 = 108 MPI tasks
  108 x 10 = 1080 total tasks

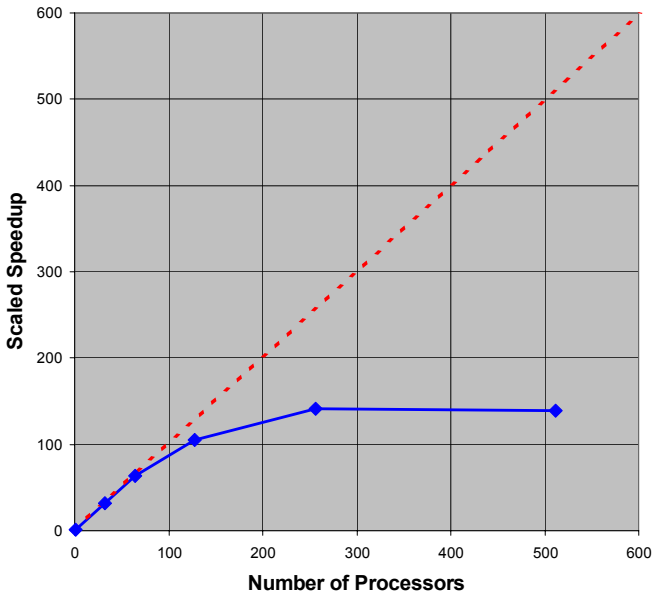
– LSF job submittal

```
bsub -n 1134 -R "span[ptile=126]" -q largeq -W 6:00 <mcnpjob
```

– MCNP5 run command in job

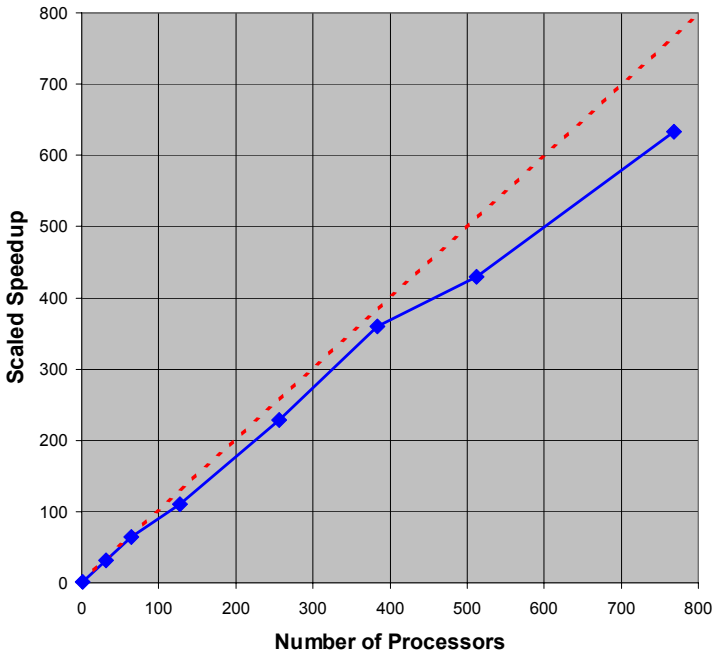
```
mpirun -np 12 mcnp5.mpi tasks 10 i=.....
```


Scaled Parallel Speedup – Eigenvalue Problem



ASCI Qsc
- MCNP5
- MPI + OpenMP
- 4 threads/node
- 1000 cycles

Scaled Parallel Speedup – Fixed-Source Problem



ASCI Qsc
- MCNP5
- MPI + OpenMP
- 4 threads/node
- 4 dumps + 16 rendezvous

References

- FB Brown & Y Nagaya, "The MCNP5 Random Number Generator", *Trans. Am. Nucl. Soc.* (Nov., 2002)
- F.B. Brown, "Random Number Generation with Arbitrary Strides," *Trans. Am. Nucl. Soc.* (Nov., 1994)
- S. Matsuura, F.B. Brown, R.N. Blomquist, "Parallel Monte Carlo Eigenvalue Calculations," *Trans. Am. Nucl. Soc.* (Nov. 1994)
- "MPI: A Message Passing Interface", http://www-unix.mcs.anl.gov/mpi/index.html
- "PVM: Parallel Virtual Machine", http://www.epm.ornl.gov/pvm
- "OpenMP Fortran Application Program Interface", http://www.openmp.org

Parallel MCNP Calculations with Windows PCs

By Forrest Brown, Tim Goorley, Jeremy Sweezy

MCNP Development Team, X-5
X-Division (Applied Physics)
Los Alamos National Laboratory

Disclaimer

- This presentation describes the MCNP5 parallel topics and capability on PCs running Microsoft Windows 9x/NT/2000/ME operating systems.
- There are a few differences between Windows and Linux, Unix, SGI, DEC, etc. Limitations or Advantages on Windows do not necessarily apply to the other platforms.

Introduction

There were earlier attempts to port PVM capabilities to PCs:

MCNP4B with pvm was successfully ported to Linux, then Windows 95 & NT 3.0 PCs. Unfortunately, the Windows port of PVM at the time had to be changed and re-compiled, which was not easy to do. These capabilities were not released to users.

Some users were able to port MCNP4C w/ PVM, but encountered similar problems.

With the update of PVM for Windows and the release of MPICH.NT, both the MPI and PVM capabilities of MCNP have been extended to Windows PCs in the general release of MCNP5. THREADS capability has not yet been extended to Windows operating systems.

Presentation Overview

- Installing MCNP executables
- Installing MPI and/or PVM
- Building parallel MCNP executables
- Building Windows PC Clusters
- Running MCNP in parallel (demo)
- Results of test cases

Installing MCNP Executables on Windows PCs

Installing MCNP executables

There are two methods for installing MCNP5 on a Windows PC:

- InstallShield® Installer – installs everything needed to start running the sequential MCNP5 executable.
  - It also modifies the environmental variables.
  - No additional software needed to install.
  - Provides parallel executables.
  - Will NOT recompile source.
- Gmake install – After the user copies the directory structure to local drive, "gmake install" will compile the source, run the test problems and summarize unexpected differences.
  - Will NOT modify environmental variables.
  - Requires previously installed Fortan Compiler and Unix shell (Cygwin).
  - Will create a MPI parallel executable, but not a PVM executable.
  - Will recompile source.

Installing MCNP Executables - InstallShield

There are two InstallShield Installers:

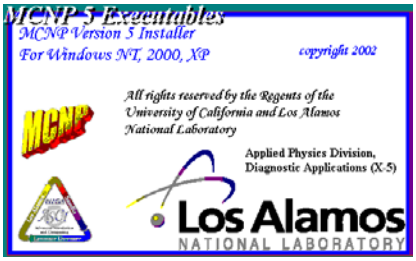
- MCNP5_Executables
  - Executables compiled with CVF
    - Mcnp5.exe – plotting & sequential executable
    - Mcnp5mpi.exe – MPI enabled executable
    - Mcnp5pvm.exe – PVM enabled executable
  - Source
  - Visual Editor
  - MCNP5 Manual and Vised Manual
  - Test Problem Suite

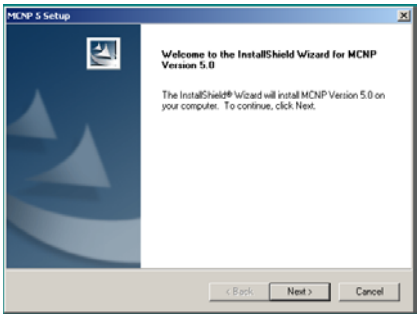
- MCNP_Data
  - Current release of data libraries
  - xsdir
  - makxsf.exe compiled with CVF
  - specs

Installing MCNP Executables - InstallShield

Typical InstallShield Process

- Start by opening "setup.exe"
- Boot screen and welcome
- Copyright Agreement
- Name, Co, Serial # (ignore)
- Select Installation Folder
  (default: Program Files\LANL\MCNP5)
  (default: Program Files\LANL\MCNPDATA)
- Installer Copies Files
- Option to Modify Env Vars
- Summary of Results
- Notice to log off and back on





Installing MCNP executables - InstallShield

The two installers will change the environmental variables:

- Executable Installer will change the environmental variables PATH and DISPLAY.
  - PATH – append the directory where mcnp executables are installed.
  - DISPLAY – set to "localhost:0" This is needed for plotting.
- Data Installer will set the environmental variable DATAPATH .
  - Set to the directory where xsdir and data libraries are stored.
  - Automatically determined when you choose to install the data libs.
- May need to log out & in for environmental variables to take effect.

Installing MCNP executables - InstallShield

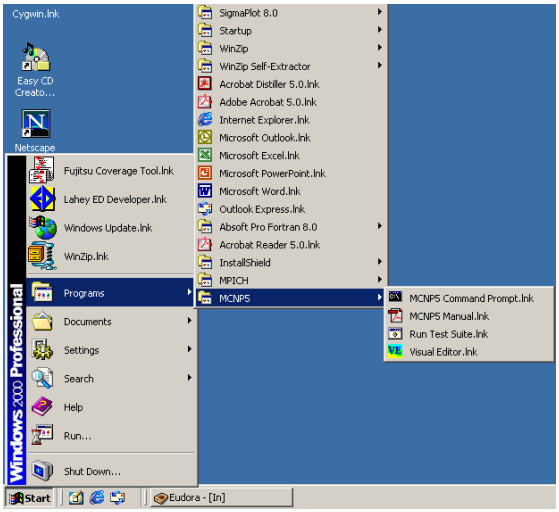
The Executable installer will also modify the Start Menu:

- A MCNP5 Command Prompt Link is created.
  - A command prompt window that opens to the directory where MCNP5 was installed.
  - User will need to cd to location of files.
- A Link to the MCNP5 Manual.pdf
  - If Adobe Reader is installed, this will automatically open a window with the Manual.
- A Link to Runprob.bat, which runs the MCNP5 test problem suite.

Installing MCNP executables - InstallShield

The InstallShield Installer will also add to the Start Menu

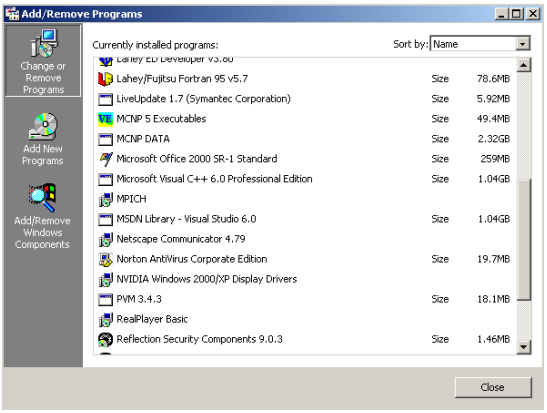
- Run the test problems
- Start a command prompt
- Open the MCNP5 Manual
  - (If Acrobat Reader is installed)
- Run the Visual Editor



UN-Installing MCNP executables - InstallShield

Uninstalling the two InstallShield packages:

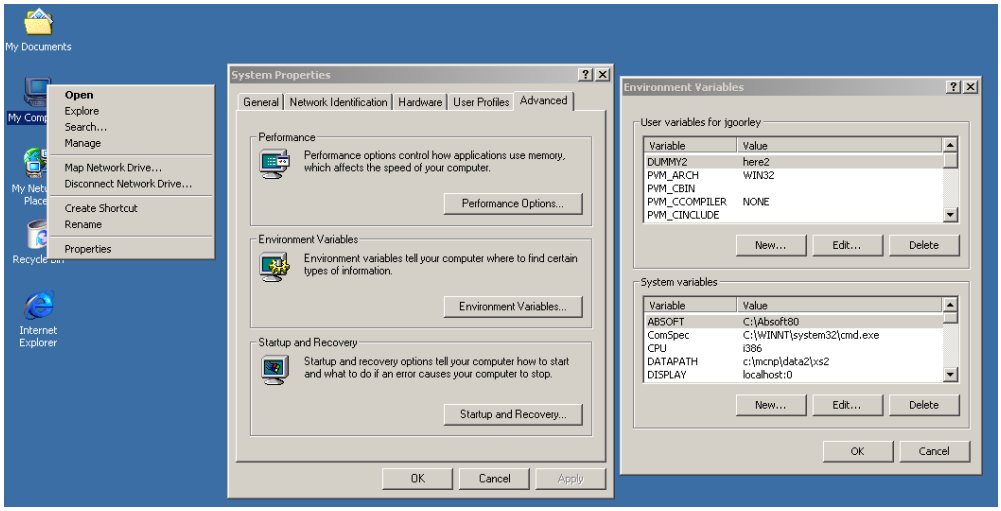
- Use the Control Panel Add/Remove
- Will not remove files that were not installed ie. the test suite output
- Remove these manually



- Will not change environmental variables
- Change manually if necessary

UN-Installing MCNP executables - InstallShield

For Windows 2000, to change the environmental variables, right click on My computer, select properties, go to "Advanced" Tab, then select Environmental Variables.



Installing MCNP executables - gmake

The second installation procedure is analogous to the installation procedure used on other platforms (Linux, Unix, etc.)

- Useful for people who want to re-compile the source code, and especially useful for those who re-compile frequently.
- Useful for people who have more experience with Unix.
- It will compile the source code, makxsf and run the test problem suite.
- It will not modify the environmental variables.
- It is based on the gmake utility

Installing MCNP executables - gmake

gmake is a tool for automating the compilation of large amounts of source code. Proper use of *make* reduces time spent compiling programs and guarantees that programs are compiled with appropriate options and linked with appropriate versions of modules and libraries. The *make* facility uses a special *Makefile* to define and describe targets, dependencies, abbreviations and macros, directory searches, and rules to automate the build process.

With the help of the *make* facility, building MCNP for a variety of hardware platforms becomes easier for the end user. The end user simply types a *make* command, optionally specifying the desired target names and configuration features. As a prelude to issuing the *make* command, an installation script queries users about the relevant characteristics of their environment, then assigns values to special variables that are used in the special *Makefile* files that appear throughout the hierarchical levels of the source distribution.

From MCNP5 Manual (Section III) - Appendix C

Installing MCNP executables - gmake

The tar file should be extracted to a desired directory. It includes the following files:

- Source Directory
  - CVF (Compaq Developer Studio Project Files)
  - X11R6 (X11 libs, dlls and include files)
  - config (files specific to different operating systems)
  - datasrc (maksxf source)
  - dotcomm (source for routines which interface with MPI or PVM)
  - src (MCNP5 source routines)
- Testing\Regression Directory
  - Inputs (input files, test xsdir, test library, test specs file)
  - Templates (the expected output files, which actual output is compared to)
- Each Directory has its own Makefile

Installing MCNP executables - gmake

These makefiles, the configuration files, and the make utility control the execution of the compilers, linkers and subsequent running of the test problems and makxsf.

- "make install testdata"
  - Preprocesses and Compiles each mcnp5 routine (sequential plotting)
  - Links all the object files to create mcnp5.exe
  - Runs test problem suite with type 1 library file
  - Displays summary of difference files
  - Preprocesses, Compiles and Links makxsf.exe
  - Runs makxsf.exe to create type 2 library file
  - Runs test problem suite with type 2 library file
  - Displays summary of difference files

- BUT ONLY IF YOU HAVE INSTALLED ADDITIONAL SOFTWARE!

Installing MCNP executables - gmake

This method requires that you previously install:

- Cygwin – A unix shell for Windows
  - http://www.cygwin.com
  - http://www.redhat.com/apps/download/
  - Should also install gmake, perl, and gcc packages.
  - Optional X11 client package – XFree86

- A Fortran 90 Compiler
  - Compaq Visual Fortran 90 (v 6.6B)
  - Lahey Fortran 95 Professional (v 5.70c)
  - Absoft Pro Fortran 95 (v 8.0)

- A C Compiler
  - GNU gcc (v 2.95.2-5 [Cygwin special])
  - Microsoft C/C++ (v 12.00.8168)
  - Fujitsu C/C++ [only with Lahey] (v 3.0)

Installing MCNP executables - gmake

To specify these compilers on the make command line, use the CONFIG keyword.

For example, to use CVF 90 and gcc to build MCNP5, type:

make build CONFIG='compaq gcc'

FORTRAN Compilers

Compaq Visual Fortran 90 -	compaq
Lahey Fortran 95 Professional-	lahey
Absoft Pro Fortran 95 -	absoft

C Compilers

GNU gcc-	gcc
Microscoft Visual C/C++	cl
Fujitsu C/C++	fcc

Installing MCNP executables - gmake

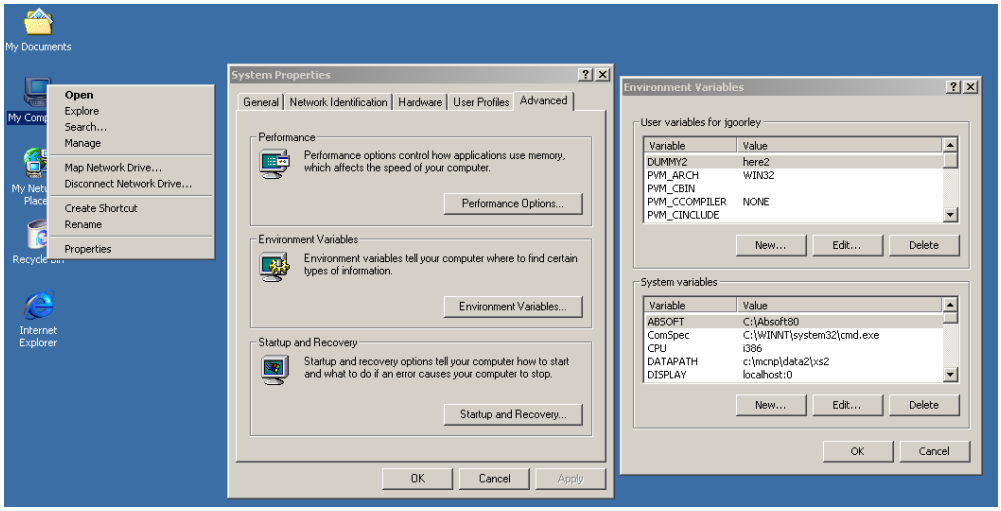
Installing with gmake will NOT change any environmental variables. This must be done manually and the method will vary depending on the Windows operating system.

- PATH – append the directory where mcnp executables are located.
- DATAPATH – set to the directory where xsdir and data libraries are stored.
- DISPLAY – set to "localhost:0" This is needed for plotting.

May require login & logout for variables to take effect.

Installing MCNP executables - gmake

For Windows 2000, to set the environmental variables, right click on My computer, select properties, go to "Advanced" Tab, then select Environmental Variables.



Installing MCNP Executable

The two installation methods are complimentary.

- Since the InstallShield method will install the source code, the user can go back and recompile the source with gmake (if you have the prerequisites for the make installation method). The user will need to manually replace the old MCNP5.exe with the newly created executable (located in the Source/src) directory.

After Installing MCNP Executables

After installing MCNP5, you should complete the following:

- The test problem suite should be run to make sure the executables are working correctly.
  - Use DOS runprob.bat (from Start Menu/MCNP5), or
  - Use Cygwin "gmake test"
  - Difference files should be zero length. If not, open them and view contents.
- Should test parallel executables as well.
  - Some tally differences can be expected. They correspond to differences in when rendezvous occur. These should not affect the final answer.

In MCTAL file:

```
vals
  9.36763E-03 0.6312  6.82551E-02 0.2590  6.31378E-02 0.3960  1.71456E-01 0.1933
  9.92486E-02 0.3608  1.29579E-01 0.5103  0.00000E+00 0.0000  5.41044E-01 0.1630
  . . .

tfc    5       1       1       1       1       1       6       8       1
       1000  6.05488E+01  4.58404E-02
       2000  6.23781E+01  3.32719E-02
       3000  6.19399E+01  2.73571E-02
       . . .
```


After Installing MCNP Executables

After installing MCNP5, you should complete the following before running the executable:

- If plotting, you MUST start the X Windows Client.
  - Reflection X (www.wrq.com/products/)
  - Exceed_NT NT (www.hummingbird.com/products/nc/exceed/index.html)
  - X-Win32 (http://www.starnet.com/)
  - XFree86 (http://www.cygwin.com/xfree)
- If running MCNP in parallel:
  - Parallel communications software MUST be installed.
  - Communications software client MUST be started.

Installing Parallel Communications Software on Windows PC

Installing Parallel Communications Software

MCNP5 for the windows can in parallel using either Parallel Virtual Machine (PVM) or Message Passing Interface (MPI) communications software. Both are freeware.

- The software must be installed and running before MCNP can work in parallel, even if a dual CPU is used.
- PVM
  - Developed at Oak Ridge National Laboratory
  - Robust Error Handling
  - Slow, Inefficient
- MPICH_NT
  - Developed at Argonne National Laboratory
  - Minimal Error Handling
  - Fast, Efficient

Installing PVM

PVM software must be installed on each computer that is going to make up the cluster.

- The Windows port of PVM is available at: http://www.csm.ornl.gov/~sscott/PVM/Software/.
- Need ParallelVirtualMachine3.4.3.zip to run and compile.

- InstallShield Installer
- Requires Administrator Privileges
- Sets variables in registry.
- DO NOT start PVM, DO NOT reboot from Installer. Quit Installer, then reboot.
- ADD ../pvm3.4/bin/win32; ../pvm3.4/lib/win32 to PATH.
- Must copy MCNP5pvm.exe to the %PVM_ROOT%/pvm3.4/bin/%PVM_ARCH% directory on all computers in the cluster.
  - %PVM_ARCH% = WIN32 on Windows PCs

Installing PVM - RSH

For Windows computers, the PVM communications software alone is not enough. A remote shell (RSH) client/server package must also be installed on each computer.

- Commercial Products available from
  - http://www.winrshd.com/
  - http://www.ataman.com/

- The permissions must be set to allow RSH or REXEC connections for the desired user accounts.

- Problems may arise if you have the same account name but with different domain names, or install PVM on the local account and then log into the domain account.

Installing PVM - RSH

Copy the ataman files to appropriate directory (c:\atrls), cd to that directory.

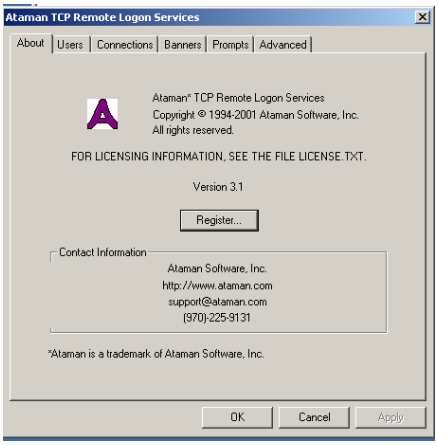
Run "atrls start install" from command prompt.

Re-boot if necessary.

Go to control panel.

Open Ataman TCP R. L. Services

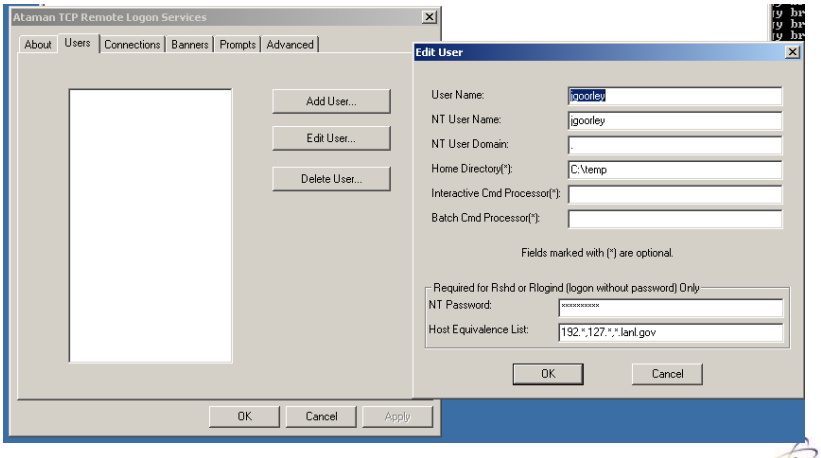
Go to Users Tab.



Installing PVM - RSH

Copy the ataman files to appropriate directory (c:\atrls), cd to that directory.

Edit User, set NT Password and/or Host Equivalence List



Installing MPICH.NT

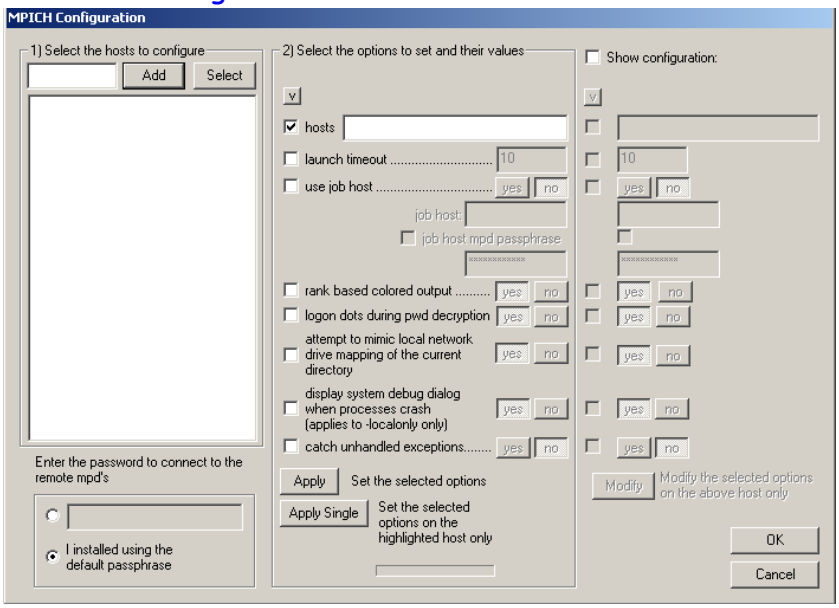
MPICH.NT software must be installed on each computer that is going to make up the cluster.

- The Windows port of MPICH is available at: http://www-unix.mcs.anl.gov/~ashton/mpich.nt/
- Need mpich.nt.1.2.4.exe to run
- Need mpich.nt.1.2.4.src.exe to compile

- InstallShield Installer
- Requires Administrator Privileges
- Install with RSH option
- MPIConfig must be run on each computer
  - Each computer's own name must be added and the settings applied.
- ADD ..\MPICH\mpd\bin to PATH
- The executable must be in the same location on all hosts.

Installing MPICH.NT

MPICH Configuration Tool



Building Parallel Executables on Windows PCs

Building Parallel Executables

While the binary executables are distributed with the InstallShield version of MCNP5, it is possible to re-compile the code if necessary.

- The InstallShield installer cannot be used to compile the code
- The gmake system can currently be used to compile the mpi executable, but not the pvm executable.
- The Compaq Developer Studio can be used to create a new MPI or PVM executable for Windows PCs.
- Corresponding PVM or MPICH.NT source must be installed.

Building Executables

There are two methods to re-compile the source.

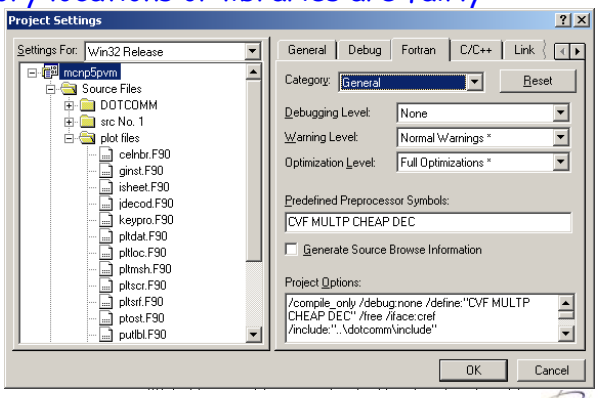
- The gmake utility
  - Command Line Based
  - Uses choice of Fortran 90/95 Compilers
  - Uses choice of C Compilers
  - Builds Sequential Plotting or Non-Plotting, or MPI Non-Plotting Executables

- The Compaq Visual Fortran Developer Studio
  - GUI Based
  - Uses CVF F90 and Microsoft C/C++ Only
  - Builds Sequential, Plotting or Non-Plotting Executables
  - Builds Non-Plotting PVM and MPI Executables

Building Parallel Executables

The CVF Developer Studio is a GUI "workspace" where it is easy to control the compile and link options. There is a CVF project for each MPI, PVM, plotting and sequential executable. The appropriate default settings for all of these projects have already been configured.

- Changes in directory locations or libraries are fairly straightforward.



Building Parallel Executables

There are a number of settings in the CVF Project which are not related to parallel calculations which may be useful to change.

- PreCompiler Settings:
  - DATE: date MCNP is compiled
  - DPATH: hard-wired data location. Can't use spaces, must use DOS name
    - ex: c:\progra~1\MCNP5\data instead of c:\program files\MCNP5\data
  - VERS: MCNP version name and number
- Compiler Settings:
  - Optimization
  - Debugging
  - Runtime error-checking
- Linker Settings:
  - Stack and Heap Settings (NO MDAS! All dynamic memory)
  - Large Address Aware

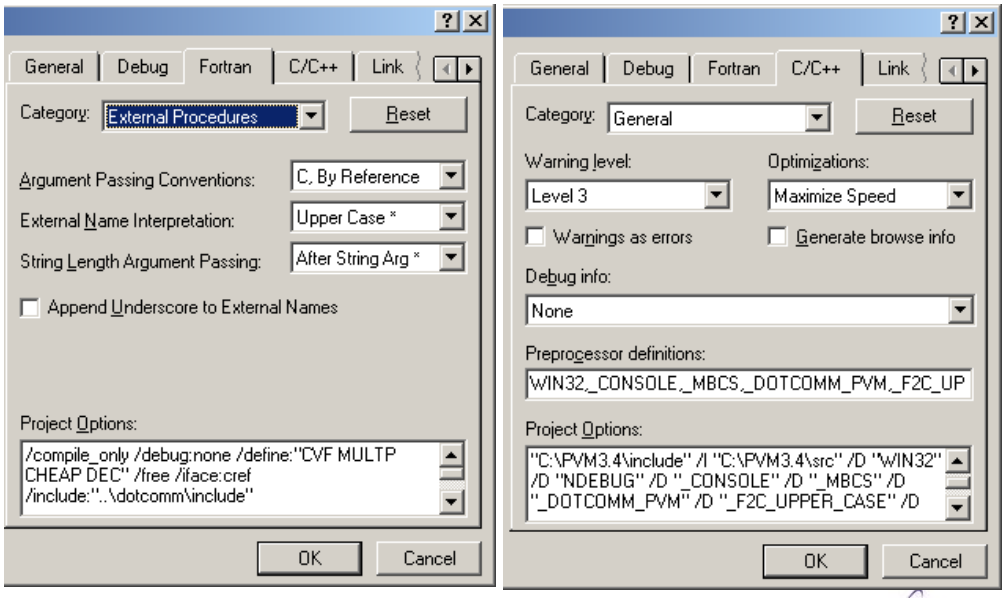
Building Parallel Executables

The CVF project settings that were changed to compile parallel executables are:

- Fortran Settings:
  - Set Preprocessor Symbols MULTP for PVM; MULTP, MPI for MPI
  - Set DMMP_NAME="mcnp5pvm" or "mcnp5mpi"
  - Argument Passing Conventions: C, By Reference
  - External Name Interpretation: Uppercase
  - String Length Argument Passing: After String Arg
  - Add ..\dotcomm\include; ..\dotcomm\src directories
- C Settings:
  - Set Preprocessor definition: _DOTCOMM_PVM or _DOTCOMM_MPI
  - Set ..\dotcomm\include,C:\PVM3.4\include,C:\PVM3.4\src (or MPI directory)
  - Set Calling Convention __cdecl
- Linkers Settings:
  - Add ws2_32.lib, libpvm3.lib, libgpvm3.lib for PVM
  - Add ws2_32.lib, mpich.lib for MPI
  - Add path to parallel library

Building Parallel Executables

In the Projects Settings Menu (Alt+F7)



Building Parallel Executables

The gmake utility can be used to build a MPI executable.
Work is in progress to allow it to build a PVM executable.

- Verify that the path to MPICH.NT .h files and library are correct in the Windows_NT.gcf files in /MCNP5/Source/config directory.
- In the /MCNP5/Source directory, type
  make clean CONFIG='compaq cl mpi'
- In the /MCNP5/Source directory, type
  make build CONFIG='compaq cl mpi'

Building Windows PC Clusters

Building Windows Clusters

May want to use parallel capabilities with:

- Regular Network Connection
- Stand-Alone Network
- Dual / Quad Processor Machines

Building Windows Clusters

For increased security, or other reasons, it is possible to build a "stand alone" network, not connected to the internet.

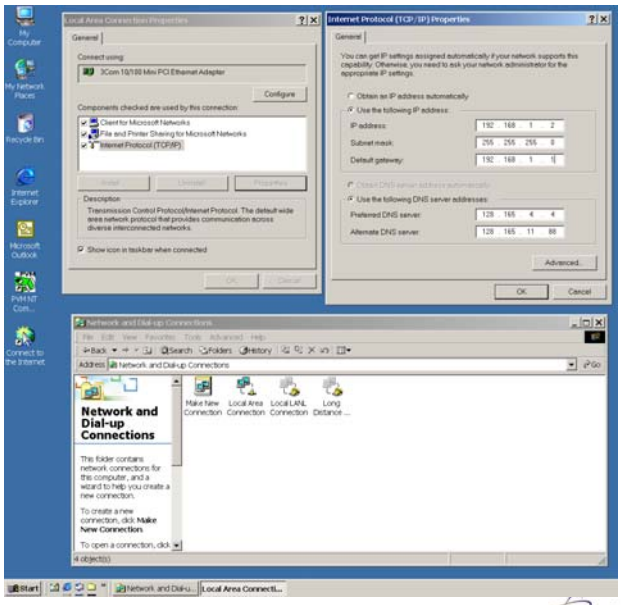
- Use Network Switch
  - 3COM Office Connect Dual Speed Switch (8 connections)
- TCP/IP Settings
  - IP Address: 192.168.1.x (where x is the switch #)
  - Subnet Mask: 255.255.255.0
  - Default Gateway: 192.168.1.1
- May need to re-boot before TCP/IP settings take effect.
- Should be able to ping with IP address.
- Possible to use hostfile? For ping and PVM, but not MPI.
  - /WINNT/System32/Drivers/etc/hosts?

Building Windows Clusters

Change TCP/IP Settings according to Windows OS. For Windows 2000:

Go to:

- Network & Dial-up Connections
- Local Area Connection
- Internet Protocol (TCP/IP) Properties
- Change IP address



Building Windows Clusters

A cluster of computers with the same operating system is preferable, but not necessary.

- PVM
  - will allow a cluster with Unix, Linux, etc. computers.
  - places some restrictions on mixing Windows 9x with NT/2000 machines.
  - MCNP restrictions on mixing Big Endian with Little Endian Architecture
- MPICH.NT
  - will NOT allow a cluster with Unix, Linux, etc. computers.
  - places some restrictions on mixing Windows 9x with NT/2000 machines.
- Note clusters can span your desktop or continents.

Building Windows Clusters

Using a dual or quad CPU PC is simple.

- You must still install PVM or MPI software
  - Do not have to install RSH Client Software for PVM
- Must start PVM or use MPIRUN
- No need to specify which CPU to use, Windows will distribute tasks across CPUs.
- Should use a negative tasks entry for PVM jobs or use MPI without BALANCE.
  - No need to use Windows Task Manager to re-prioritize tasks
- No shared memory capability for MCNP, i.e. no "threads"

Building Windows Clusters

It is convenient to have a file share across the windows cluster, so that you can easily copy installation files between them. For a stand-alone cluster:

- Set IP addresses – 192.168.1.x
- On one host (ex. 192.168.1.1)
  - Select Folder you wish to share (or create a new one)
  - Folder properties (right click)
  - Share this folder – name "Hello" properties – (ex. Everyone – Full control)
- On other PCs
  - Go to My Network Places
  - Add Network Place
  - Location of the network place – \\192.168.1.1\Hello
  - Enter a name – Hello on 192.168.1.1
  - Folder will open in new window and remain in my network places

Running MCNP in Parallel on Windows PC Clusters

Running Parallel MCNP5 - PVM

- Start PVM (c:> pvm)
- Add hosts to parallel machine from pvm>
- Start new command shell
- Cd in this new command shell to the directory you want to create your output files. Only one set of output files is created, and they are on the host where the MCNP5pvm master was started.
- Start MCNP5 run

Running Parallel MCNP5 - PVM

- Mcnp5pvm inp=test … tasks #
  - Where # is the number of spawned tasks. (Should be equal to the number of CPUs)
  - Tasks are spawned according to the order of the hosts in the virtual machine on the machine where the mcnp5pvm job is started.
  - The first machine is skipped (since the master is running on it), and tasks are created incrementally though the host list until all the possible tasks are spawned.
  - Note this may cause over subscription. MCNP will warn if the # of tasks does not equal the number of hosts in the Virtual Machine.
  - Thus it is possible to have a mixed cluster of dual and single processor PCs, and effectively use all the processors. (by adding the dual processor hosts to the virtual machine first)

Running Parallel MCNP5 - PVM

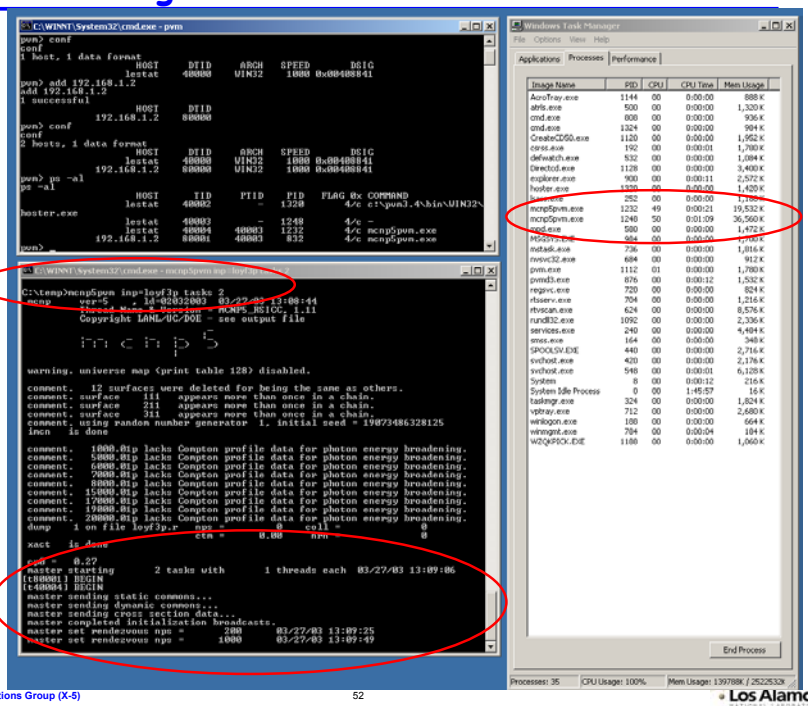
- Mcnp5pvm inp=test … tasks #
  - A positive entry for tasks does some load balancing by testing 200 particle histories on each CPU. The number of histories each task runs is then determined by the relative speed of each task.
    - Master task consumes significant CPU time, may lower with Task Manager.
    - Best option for heterogeneous clusters.
  - A negative tasks entry assumes a homogeneous cluster and each spawned task runs the same number of tasks.
    - If any CPU is over-subscribed, delays will occur.
    - Best option for homogeneous clusters.

Running Parallel MCNP5 - PVM

PVM Prompt shows configuration & processes on all hosts

Regular command prompt to start mcnp job

Task Manager (right) shows processes on this host and CPU utilization



Running Parallel MCNP5 - MPI

- Start command shell
- Cd in this command shell to the directory you want to create your output files. Only one set of output files is created. They correspond to the working directory of the process zero task.
  - Note, it is possible to start a process zero task on a different host than where the mpirun command is issued.
- Start MCNP5 run
- Need to pass environmental variables for DATAPATH. Since mpirun does not pass these variables to process zero, if DATAPATH is not hardwired correctly or in the directory where the input deck starts up, will not find the file xsdir!

Running Parallel MCNP5 - MPI

- Since environmental variables are not passed, need to specify location of executable.
  - mpirun –np 3 c:\progra~1\LANL\MCNP5\mcnp5mpi inp=test
  - NOTE short form of c:\Program Files\ path
- Can copy mcnp5mpi.exe to current directory, but also need to copy to same location on all hosts.
  - mpirun –np 3 mcnp5mpi inp=test
- The first time you run, you will need to enter account name and password.

Running Parallel MCNP5 - MPI

- Mpirun –np # mcnp5mpi inp=test
  - The –np # option specifies the total number of tasks, including the master.
  - The –np # should equal the number of CPUs +1
  - This option assumes the localhost, used for DUAL or QUAD PCs.
- Mpirun –hosts # name1 #1 name2 #2 … mcnp5mpi inp=test
  - The –hosts options allows you to specify the number of machines in the cluster and the number of processes created on each machine.
  - Useful for a mixture of single, dual and quad processor machines
  - The default is to assume a homogeneous cluster and not load balance
- Mpirun –hosts # name1 #1 … mcnp5mpi inp=test BALANCE
  - Does load-balancing in the same way that PVM does.
  - Useful for a mixture of processor speeds

For more Mpirun options see the MPICH.NT Users Manual

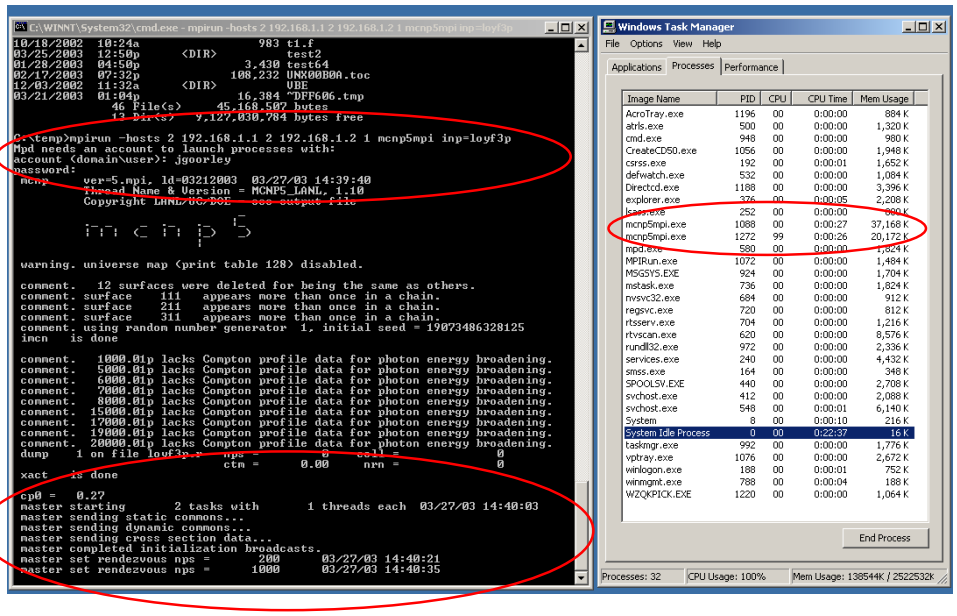
Running Parallel MCNP5 - MPI

As an alternative to the specifying options on the command line, it is possible to create a configuration file.

exe c:\progra~1\LANL\MCNP5\mcnp5mpi
env DATAPATH=c:\progra~1\LANL\MCNPDATA
dir c:\workingdir
hosts
Computer1_name #processes
192.168.1.2 #processes
192.168.1.3 #processes

Can use machine file options to specify different directories on different computers? – See MPI Manual or MPIRUN.

Running in Parallel - MPI



Running Parallel MCNP5

- Control of Master-Slave task communication controlled by the 5th entry of the prdmp card. For multiprocessing:
- A negative entry means rendevious every 1000 particles
- A zero entry means rendezvous 10 times in the run, rounded to the nearest 1000 particles.
- A positive entry means rendezvous after that many particles.
- Setting this number to the NPS will minimize communication during the run.

Running Parallel MCNP5 - Output

Both PVM and MPI MCNP jobs have similar screen output:

```
dump   1 on file loyf3r.r  nps =       0     coll =          0
                           ctm =    0.00     nrn =          0
xact   is done

cp0 =   0.27
master starting      2 tasks with      1 threads each  03/19/03 15:06:03
master sending static commons...
master sending dynamic commons...
master sending cross section data...
master completed initialization broadcasts.
master set rendezvous nps =      200     03/19/03 15:06:20
master set rendezvous nps =     1000     03/19/03 15:06:34
master set rendezvous nps =     2000     03/19/03 15:07:15
master set rendezvous nps =     3000     03/19/03 15:07:57
master set rendezvous nps =     4000     03/19/03 15:08:46
```


Running Parallel MCNP5 - Output

Both PVM and MPI MCNP jobs have similar file output:

```
 estimated system efficiency:  net = 89%  loss =  9%
  (locks) +  1% (comm.) +  1% (misc.)


 number of histories processed by each task
         0      3470      1530
```

First # – Master – should be zero unless spawned task dies.

Following numbers are for subtasks.

PVM and MPI assigns process numbers differently, so they may be reversed from each other.

Running Parallel MCNP5

Dual CPU desktop Timing Study

Precision 520 (Dual 2.0 GHz Pentium Xeon® Processors, 768 Megabytes RAM, 512 kbytes L2 cache, 100 MHz bus) running Windows 2000.

Wall Clock Runtimes (min:sec)	Sequential	PVM tasks 2	PVM tasks -2	PVM* tasks 2	PVM tasks 3	MPI 3 processes
Nps 10,000	7:36	9:42	5:38	4:47	7:15	4:18
Nps 100,000	72:34	90:55	41:24	40:37	54:13	38:44

* Indicates the slave task's priority was changed to "above normal" with the Windows Task Manager

Running Parallel MCNP5

Small Laptop Cluster Timing Study

DELL Inspiron 8200

Pentium IV ®, 1.6 GHz, 1024 Mbytes RAM, 512 kbytes L2 Cache

DELL Lattitude C800

Pentium III®, 1.0 GHz, 512 Mbytes RAM, 256 kbytes L2 Cache

Wall Clock Runtimes (min:sec)	Sequential		PVM tasks 2	PVM* tasks 2	MPI 3 processes	MPI 3 processes BALANCE
Task Distribution	Pentium 4	Pentium 3	P4:Master +Slave P3:Slave	P4:Master +Slave P3:Slave	P4:Master +Slave P3:Slave	P4:Master +Slave P3:Slave
NPS 10,000	9:41	30:25	11:41	10:05	16:33	9:30
NPS 100,000	90:55	298:54	143:32	83:27	153:29	75:34

Conclusions

- Test a sample job on your own cluster:
- Short Jobs have high % overhead, inefficient
- Master task may be using CPU time, inefficient
  - May use Task Manage to improve efficiency
- MPI is more efficient on a Dual Processor than PVM
- MPI w/ BALANCE is more efficient on a Heterogeneous Cluster than PVM.

Trouble-Shooting

MPI Trouble-Shooting

- Can you ping all computers in cluster?
  - Network Problem
- Can you run mpirun –localhost –np 3 mcnp5mpi?
  - Logged into local (not domain) account?
  - Path not defined?
  - Did you run MPI CONFIG?
  - May need to reinstall MPI.
- Can you start MCNP5mpi?
  - MCNP5mpi not compiled with MPI enabled?
  - MCNP5mpi not in same directory on all computers?
  - Naming problem TCP/IP Address vs. computer name (hostfile)

PVM Trouble-Shooting

- Can you ping all computers in cluster?
  - Network Problem
- Can you "rsh host dir" to all computers in cluster?
  - RSH permissions problem, check RSH permissions
  - Domain Account Problem?
- Can you add host at pvm prompt?
  - Domain vs Local Account Problem?
  - pvm> add "hostname lo=name so=pw"
  - Re-install PVM?
- Can you run hello from command prompt and get replies from all computers?
  - Use "conf" at pvm prompt to view hosts
- Can you start MCNP5pvm with tasks entry?
  - MCNP5pvm not compiled with PVM enabled
  - MCNP5pvm not in %PVM_ROOT%/bin/Win32 directory on all computers


MCNP5
&
PC Clusters - Linux

Forrest Brown, Tim Goorley, Jeremy Sweezy
MCNP Development Team, X-5
X-Division (Applied Physics)
Los Alamos National Laboratory

M&C 2003 Gatlinburg, TN
April 11, 2003


Disclaimer

- This presentation describes MCNP5 parallel topics and capability on PCs running the Linux operating system.
- These instructions are targeted to i386 systems running RedHat Linux, but should be applicable to other hardware and other flavors of Linux.
- The % symbol denotes the command line.
- The `fixed width font` is used to denote commands to be entered on the command line, executables, files, and file entries.

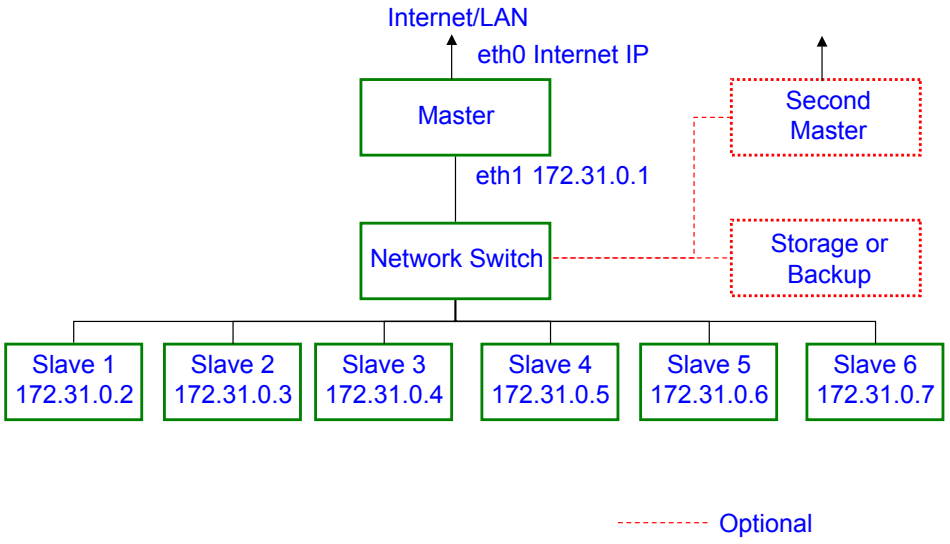
Presentation Overview

- Overview of Linux clusters
- Diskless Linux clusters
  - Building diskless Linux clusters
  - Diskless Linux cluster boot demo
- **MCNP5 on Linux Clusters**
  - Installing MPI and PVM
  - Building parallel MCNP executables
  - MCNP5 Parallel Calculations
  - Running MCNP in parallel (demo)

Linux Clusters – Why?

- Stability
- Flexibility
- Linux supports almost all hardware
- Commercial and free distributions of Linux OS
- Large user community
- Network booting capability
- Supports various types of network file systems (NFS, PVFS, etc)
- Cost

Linux Clusters - Topology



Linux Clusters

Private IP Addresses:

- The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private internets:
  - 10.0.0.0 - 10.255.255.255
  - 172.16.0.0 - 172.31.255.255
  - 192.168.0.0 - 192.168.255.255

Linux Clusters

Clustering Software:

- Scyld Beowulf (www.scyld.com)
  (www.linuxcentral.com has Scyld CD for $2.95 w/o support)
- NPACI Rocks (www.rocksclusters.org)
- OSCAR (oscar.sourceforge.net)
- Do-it-yourself
  - Standalone Slaves
  - Diskless Slaves

Linux Clusters

Job Scheduling Software:

- Maui Scheduler (www.supercluster.org/maui)
- OpenPBS(www.openpbs.org)
- PBSPro(www.pbspro.com)
- LSF (www.platform.com)

Diskless Linux Clusters

- Advantages
  - Easy setup
  - Little maintenance required for the slaves nodes.
  - Slave nodes can be added and replaced rapidly.
  - Ad-hoc clusters can be assembled rapidly.
  - Reduced cost of slaves.

- Disadvantages
  - Complete operating system resides on the network (slower).
  - No local disk for swap space.
  - Complete cluster reliant on the master.

Diskless Linux Clusters

To Build a Diskless Linux Cluster you need:

1. One computer as the master host running Linux
   - Linux operating system
   - Two network cards (only one required if no external network)
   - Video card
   - Monitor and keyboard
   - Hard drives
   - CD-ROM and Floppy drives

2. Some type of network
   - Network hub or network switch

3. One or more computers as slaves
   - One network card per slave
   - No hard drive required
   - No operating system required
   - Video card
   - CD-ROM or Floppy drives

4. Optional
   - KVM switch

Diskless Linux Clusters

Building a Diskless Linux Cluster Overview:

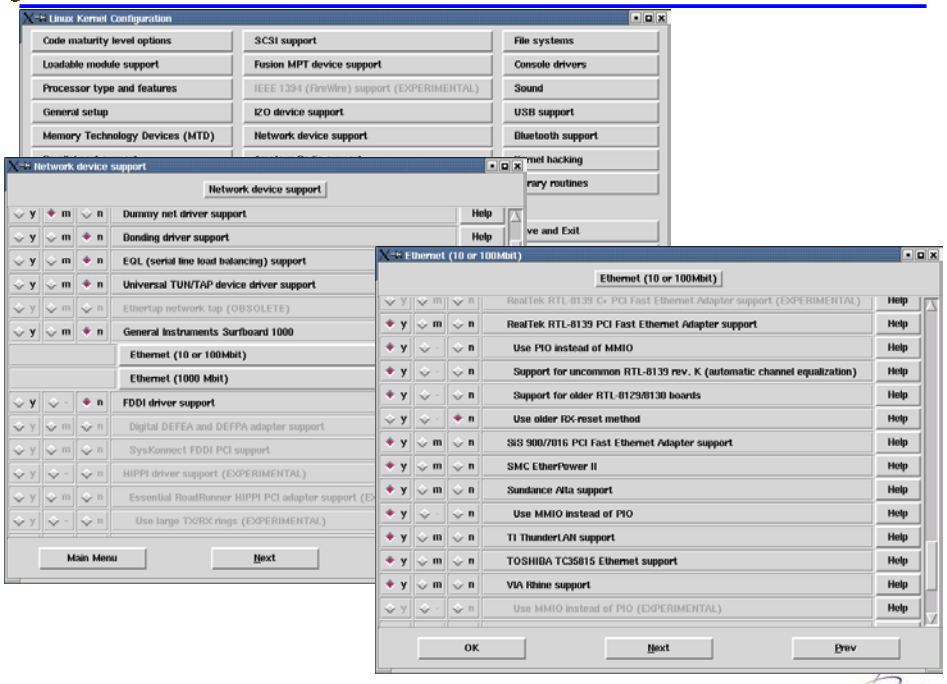
1. Create slave boot media
   - Build Linux kernel for the slaves
   - Install and configure boot loader (syslinux) on a floppy or CD.
   - Install Linux kernel on to the floppy or CD.
2. Create slave file system
3. Configure NFS and security
4. Connect the slaves to the master and boot

Diskless Linux Clusters

Build a kernel for the Diskless Slaves:

1. Download latest Linux kernel (www.kernel.org)
2. Unpack in `/usr/src/linux`
3. `% make mrproper; make xconfig`
4. Select the following options to be build into the linux kernel.
   1. Networking options ---> IP: kernel level autoconfiguration
   2. File Systems ---> Network File Systems ---> NFS file system support ---> NFS file system support
   3. File Systems ---> Network File Systems ---> NFS file system support ---> Root over NFS
   4. Block devices ---> Loopback device support
   5. Network device support ---> Ethernet (10 or 100 Mbit) ---> Select your network cards
5. `% make dep; make bzImage`
6. The new kernel resides at
   `% /usr/src/linux/arch/i386/boot/bzImage`

Diskless Linux Clusters



Diskless Linux Clusters

Create slave boot media:

1. Insert a disk into the floppy drive and run:
   - `% syslinux -s /dev/fd0`

2. Mount the floppy and edit the file `syslinux.cfg`
   (note that the "append" line shown here as 3 lines is really 1 long line)

```
default linux
append init=/sbin/init root=/dev/nfs
   ip=172.31.0.2:172.31.0.1:172.31.0.1:255.255.255.0:slave1:eth0:'bootp'
   nfsroot=172.31.0.1:/tftpboot/172.31.0.2
Say Remote Booting Slave1
Say Slave1 IP Address = 172.31.0.2
```

3. Copy the linux kernel to the floppy and rename to "linux"

```
% cp /usr/src/linux/arch/i386/boot/bzImage /mnt/floppy/linux
```

Diskless Linux Clusters

Create first slave node file system:

1. On the master node
   ```
   % mkdir /tftpboot; cd /tftpboot
   ```

2. Download and run the `nfsrootinit` script to create the first root file system. (http://etherboot.sourceforge.net/doc/html/nfsrootinit.txt)
   ```
   % chmod u+x nfsrootinit.txt
   % ./nfsrootinit.txt 172.31.0.2
   ```

3. Edit `/tftpboot/172.31.0.2/etc/fstab` to mount the correct directories via NFS.
   ```
   None    /dev/pts    devpts      gid=5,mode=620  0 0
   None    /proc       proc        defaults        0 0
   None    /dev/shm    tmpfs       defaults        0 0
   172.31.0.1:/tftpboot/172.31.0.2 /       nfs rw,soft,rsize=8192,wsize=8192,intr
   172.31.0.1:/home                /home   nfs rw,soft,rsize=8192,wsize=8192,intr
   172.31.0.1:/usr                 /usr    nfs rw,soft,rsize=8192,wsize=8192,intr
   ```

4. Edit `/tftpboot/172.31.0.2/sysconfig/network` and change the HOSTNAME variable.
   ```
   HOSTNAME="slave1.mcnpengine.lanl.gov"
   ```


Diskless Linux Clusters

Duplicate slave node file system for other slaves:

1. Download and run the `nfsrootdup` script to duplicate the first root file system. (http://etherboot.sourceforge.net/doc/html/nfsrootdup.txt)
   ```
   % chmod u+x nfsrootinit.txt
   % ./nfsrootinit.txt 172.31.0.2 172.31.0.3
   ```

2. Edit `/tftpboot/172.31.0.3/etc/fstab` to mount the correct directories via NFS.
   ```
   None    /dev/pts    devpts      gid=5,mode=620  0 0
   None    /proc       proc        defaults        0 0
   None    /dev/shm    tmpfs       defaults        0 0
   172.31.0.1:/tftpboot/172.31.0.3 /       nfs rw,soft,rsize=8192,wsize=8192,intr
   172.31.0.1:/home                /home   nfs rw,soft,rsize=8192,wsize=8192,intr
   172.31.0.1:/usr                 /usr    nfs rw,soft,rsize=8192,wsize=8192,intr
   ```

3. Edit `/tftpboot/172.31.0.3/sysconfig/network` and change the HOSTNAME variable.
   ```
   HOSTNAME="slave2.mcnpengine.lanl.gov"
   ```

Diskless Linux Clusters

Set up the NFS Server and Security:

1. Edit /etc/exports
```
/home     172.31.0.0/255.255.255.0(rw,no_root_squash)
/usr      172.31.0.0/255.255.255.0(rw,no_root_squash)
/tftpboot 172.31.0.0/255.255.255.0(rw,no_root_squash,no_subtree_check)
```

2. Start or restart the NFS server. On a Redhat system use:
```
% /etc/init.d/nfs start
```
or
```
% /etc/init.d/nfs restart
```

3. Add the slaves hosts to your /etc/hosts file
```
172.31.0.2     slave1.mcnpengine.lanl.gov         slave1
172.31.0.3     slave2.mcnpengine.lanl.gov         slave2
172.31.0.4     slave3.mcnpengine.lanl.gov         slave3
172.31.0.5     slave4.mcnpengine.lanl.gov         slave4
172.31.0.6     slave5.mcnpengine.lanl.gov         slave5
```

4. Edit /etc/hosts.allow and add the following line
```
all : 172.31.0.
```

5. If the master is connected to an external network use a firewall (iptables or ipchains) to block access to all but a limited number of privileged ports.

Linux Clusters – Information

- Books
  - "Linux Clustering: Building and Maintaining Linux Clusters" Charles Bookman
  - "Beowulf Cluster Computing with Linux" Thomas Sterling, Editor
- Websites
  - www.beowulf.org
  - www.beowulf-underground.org
- HOWTOs (http://www.tldp.org/)
  - Beowulf-HOWTO
  - Linux Cluster HOWTO
  - Diskless Nodes HOWTO
  - Root over NFS Clients & Server HOWTO
  - Root over NFS - Another Approach HOWTO
  - Network Boot and Exotic Root HOWTO
  - NFS-Root-Client Mini-HOWTO
  - NFS-Root Mini-HOWTO

Diskless Cluster Demo

1. Boot master node.
2. Boot slave nodes.
3. Verify network using ping.

Installing MCNP5 on Linux Clusters

- Fortran 90 Compilers
  - Absoft 8.0 QF3
  - Lahey 6.1e
  - Portland Group Compiler 4.0-2
- GCC Compiler
- MPICH 1.2.5 (www.mcs.anl.gov/mpi/mpich)
- PVM 3.4.4 (www.csm.ornl.gov/pvm/pvm.html)
- The MCNP installation procedure is documented in Appendix C of the MCNP5 manual.

Installing MCNP5 on Linux Clusters

Setting Environment Variables for MPICH compilation:

1. Set the FC and F90 environment variables to match your Fortran compiler

   Absoft

```
% export FC="f77 -YEXT_NAMES=LCS -YEXT_SFX=_"
% export F90="f90 -YEXT_NAMES=LCS -YEXT_SFX=_"
```

   Lahey

```
% export FC="lf77"
% export F90="lf90"
```

   Portland

```
% export FC="pgf77 -tp px -L/usr/local/pgi/linux86/lib -lpgftnrtl -lpgc"
% export F90="pgf90 -tp px -L/usr/local/pgi/linux86/lib -lpgftnrtl -
lpgc"
```

2. Set the CC environment variable

```
% export CC="gcc"
```


Installing MCNP5 on Linux Clusters

Compiling MPICH:

1. Download MPICH and unpack

2. Run the configure script

```
% ./configure --prefix=/usr/local/mpich-1.2.5 –with-device=ch_p4
--with-comm=shared --with-arch=LINUX >& configure.log
```

3. Run make

```
% make >& make.log
% make install >& install.log
```

4. Add all nodes in your cluster to

```
% /usr/local/mpich-1.2.5/share/machines.LINUX
```

5. Modify your path to include

```
% /usr/local/mpich-1.2.5/bin
```

Installing MCNP5 on Linux Clusters

Running Parallel MCNP5 – MPI:

Use the following commands to start an MCNP5 job using MPICH

- `% mpirun -np # mcnp5.mpi inp=test eol`
  - `#` = number of MPI processes
  - `eol` instructs MCNP to ignore all following commands, including those added by MPICH.
  - Useful for a cluster with identical processors.
- `% mpirun -np # mcnp5.mpi inp=test balance eol`
  - `balance` keyword provides for dynamic load balancing
  - Useful for a cluster with a mixture of different speed processors or a cluster with varying loads.

Installing MCNP5 on Linux Clusters

Notes on using MPICH with MCNP5:

- With mpich-1.2.4 support for the POSIX sched_yield call was added.
  - This significantly increases the performance when -comm=shared is chosen, when there are more total processes (including operating system and other user processes) than processors on a node. This is now the default on Linux. However, when there are fewer processes than processors, lower latencies can be achieved by configuring with --disable-yield. Enable with --enable-yield=sched_yield.
  - This feature should be enabled on small clusters but probably should not be enabled for very large clusters.
  - To use the POSIX sched_yield function use a 2.4.20 or greater Linux kernel.

Installing MCNP5 on Linux Clusters

Notes on using MPICH with MCNP5:

- To use of ssh instead of rsh
  - `% P4_RSHCOMMAND="ssh -x"`

  - Create your ssh authentication key.
    `% ssh-keygen`
    This will generate a private/public key pair. The private key will be saved in `~/.ssh/identity` and the public key will be saved in `~/.ssh/identity.pub`

  - Authorize Access. Place your public key in your `~/.ssh/authorized` keys file.
    `% cat ~/.ssh/identity.pub >> ~/.ssh/authorized_keys`

  - In order to avoid typing in your pass phrase each time ssh is invoked, an ssh-agent needs to be created and your pass phrase added.
    `% ssh-agent $SHELL`
    `% ssh-add`

Compiling PVM

1. Download and unpack PVM into its final location
   `% cd /usr/local/; tar zxvf pvm3.4.4.tgz; cd /usr/local/pvm3`

2. Set the PVM environment variables in your `.bashrc` file
   ```
   export PVM_ROOT=/usr/local/pvm3
   export PVM_ARCH=LINUX
   ```

3. Source the `.bashrc` file
   `% source ~/.bashrc`

4. Make PVM
   `% make`

5. Modify your path to include
   ```
   /usr/local/pvm3/lib
   /usr/local/pvm3/lib/LINUX
   /usr/local/pvm3/bin/LINUX
   ~/pvm3/bin/LINUX
   ```

Starting PVM

- To start PVM, run `$PVM_ROOT/lib/pvm`. This starts the PVM console.
  ```
  % pvm
  ```
- More hosts can be added to your "virtual machine" by using the console "add" command.
  ```
  pvm > add hostname
  ```
- To add a list of hosts, use the hostfile option. List the hostnames in a file and start pvm with the filename as an argument
  ```
  % pvm hostfile
  ```
- To display the current virtual machine configuration
  ```
  pvm > conf
  ```
- To exit the PVM console but leave PVM running
  ```
  pvm > quit
  ```
- To stop PVM
  ```
  pvm > halt
  ```


Running MCNP5 with PVM

Notes on using PVM with MCNP5:

- After compilation of MCNP5 with PVM support copy the mcnp5.pvm executable to either
  - `$HOME/pvm3/bin/LINUX/mcnp5.pvm`
    or
  - `$PVM_ROOT/bin/LINUX/mcnp5.pvm`
- When using PVM with load balancing the master process does not yield the CPU. This can be accomplished by hand with the `renice` command.

Running MCNP5 with PVM

Running Parallel MCNP5 – PVM:

Use the following commands to start an MCNP5 job using PVM

- `% mcnp5.pvm inp=test tasks -#`
  - The the negative sign before the number of PVM slave processes turns off dynamic load balancing.
  - # = number of PVM slave processes
  - Useful for a cluster with identical processors.
- `% mcnp5.pvm inp=test tasks #`
  - The lack of the negative sign before the number of PVM slave processes provides for dynamic load balancing.
  - Useful for a cluster with a mixture of different speed processors or a cluster with varying loads.

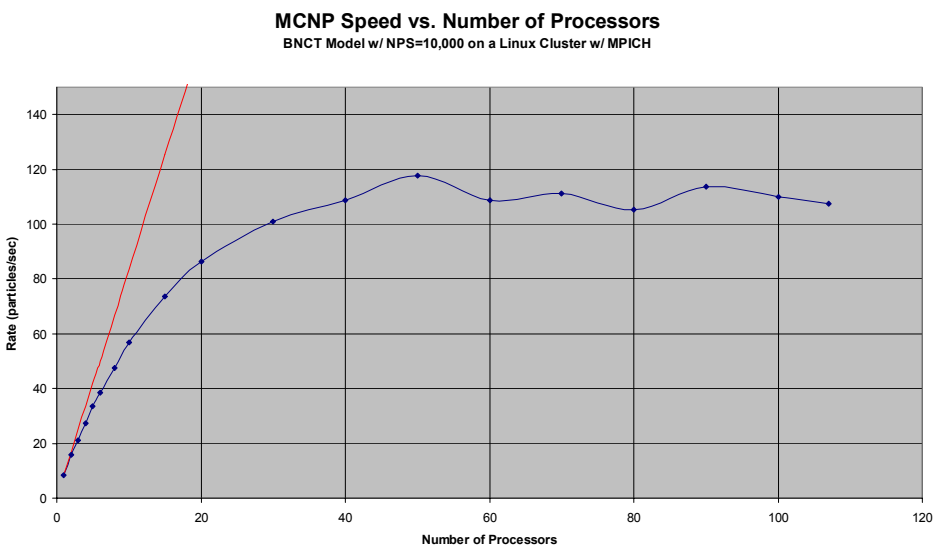
MCNP5 Parallel Calculations

- Dual CPU Desktop Timing Study
  - Dual 2.2GHz Intel Pentium IV XEON CPUs, 1 GB RAM, 512k L2 cache, running Linux 2.4.20 kernel and Redhat Linux 7.3 distribution

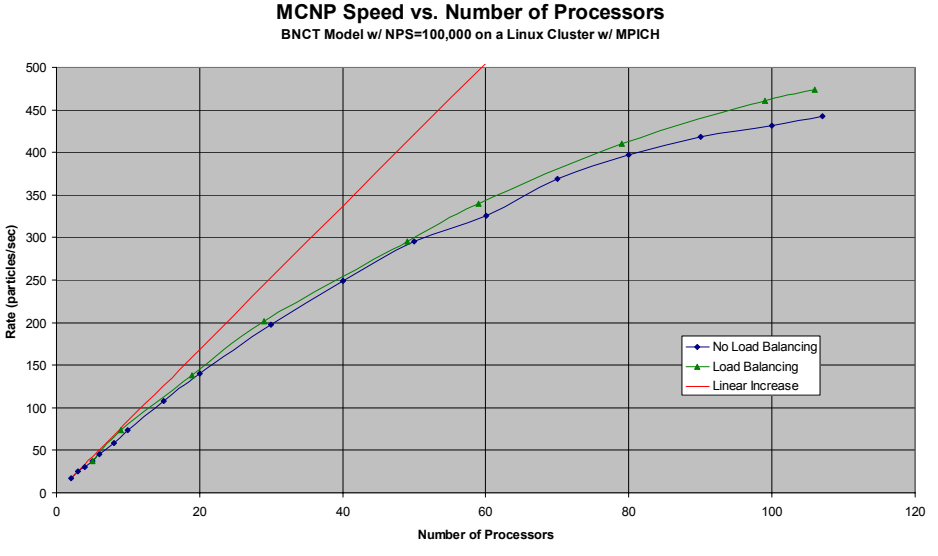
Wall Clock Runtimes	Sequential	PVM tasks 2	PVM tasks -2	MPI -np 3	MPI -np 3 balance
NPS 10,000	9:41	6:09	5:12	5:21	5:11
NPS 100,000	100:49	58:42	49:32	52:14	48:49

Using:

- Type 1 cross sections
- MPICH 1.2.5 compiled with --enable-yield=sched_yield
- PVM 3.4.4



MCNP5 Parallel Calculations



Conclusions

- Load balancing provides increased efficiency for small heterogeneous clusters and for large homogenous clusters.
- Short jobs have high % overhead reducing the effectiveness of using more processors.
- Master task may be using CPU time, inefficient
  - For MPICH use --enable-yield=sched_yield
  - For PVM use `renice` to lower master priority

Demo of MCNP5 on a diskless laptop cluster running Linux