ETNUS
TOTALVIEW

Blue Gene/L Workshop

Reno, NV

October, 2003

TotalView on Blue Gene/L

John DelSignore
Chief Architect

Outline of Today's Talk

- About Etnus TotalView
- Porting TotalView to BG/L
- Scalability and Performance
- Do We Need a Paradigm Shift?
- Questions & Answers

ETNUS TOTALVIEW

About Etnus TotalView

- World’s leader in UNIX/Linux debugging solutions
- Parallel Processing, SMP and Distributed Systems
  - Multi-process and multi-thread support
  - Distributed, clusters, MPI, pthreads, OpenMP
- Customers include
  - Major government research and academic labs worldwide
  - Software development companies
  - Computer hardware vendors
  - Companies in Finance, Entertainment, Telecommunications, Energy, Aerospace, Climate Modelling, Automotive
- TotalView has been continuously developed and extended for 17 years
- We’re proud to be the ASCI debugger of choice

Platforms

- Present
  - IBM AIX Power (RS6000, SP)
  - HP/Compaq Tru64 Alpha
  - Intel Linux X86 and IA64
  - SGI IRIX MIPS
  - Sun Solaris SPARC
- Future
  - AMD Linux X86_64
  - HP HPUX IA64
- Past
  - Compaq Linux Alpha
  - HP HP-UX PA-RISC
  - Many, many others
- Fujitsu, NEC, Cray, Hitachi
- Red Storm, Blue Gene/L

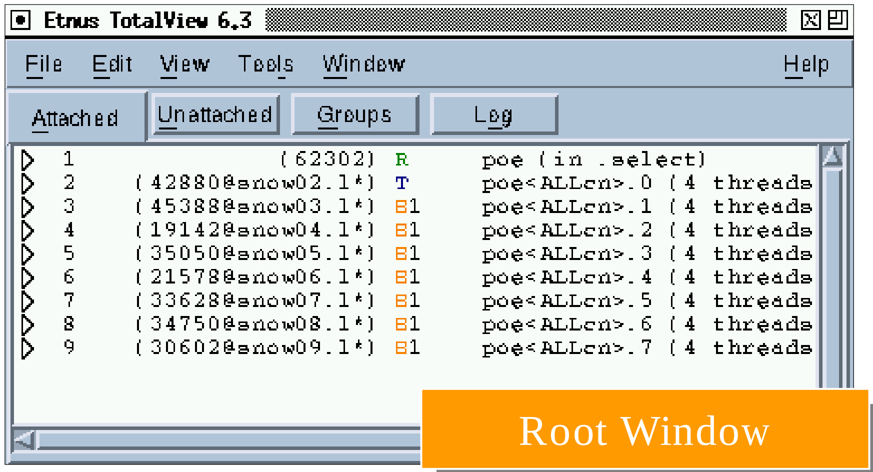
Languages

- C/C++
- Fortran 77/90/95
  - F90 types
  - Modules
  - By-descriptor arrays
- UPC
- Mixed Languages
- Assembly
- Mixed Java/C/C++
  - with the CodeRoad JNI Bridge

Supports Leading Compilers

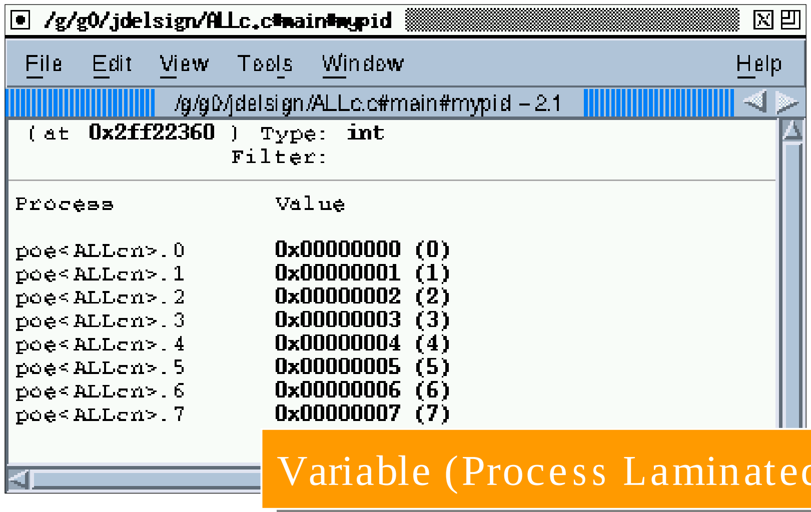
- IBM XL/VA
- GCC 3
- SGI MIPS Pro
- Sun ONE Studio
- Intel C/C++ for Linux
- Intel Fortran for Linux
- Intel KCC
- Intel KAI Guide
- PGI 3
- Lahey/Fujitsu
- Apogee
- HP/Compaq C/C++/F90
- Compaq UPC 2.0

TotalView GUI & CLI


Etnus TotalView 6.3
File Edit View Tools Window Help
Attached Unattached Groups Log

```
1                (62302)  R    poe (in .select)
2    (42880@snow02.1*)    T    poe<ALLcn>.0 (4 threads
3    (45388@snow03.1*)    B1   poe<ALLcn>.1 (4 threads
4    (19142@snow04.1*)    B1   poe<ALLcn>.2 (4 threads
5    (35050@snow05.1*)    B1   poe<ALLcn>.3 (4 threads
6    (21578@snow06.1*)    B1   poe<ALLcn>.4 (4 threads
7    (33628@snow07.1*)    B1   poe<ALLcn>.5 (4 threads
8    (34750@snow08.1*)    B1   poe<ALLcn>.6 (4 threads
9    (30602@snow09.1*)    B1   poe<ALLcn>.7 (4 threads
```

Root Window

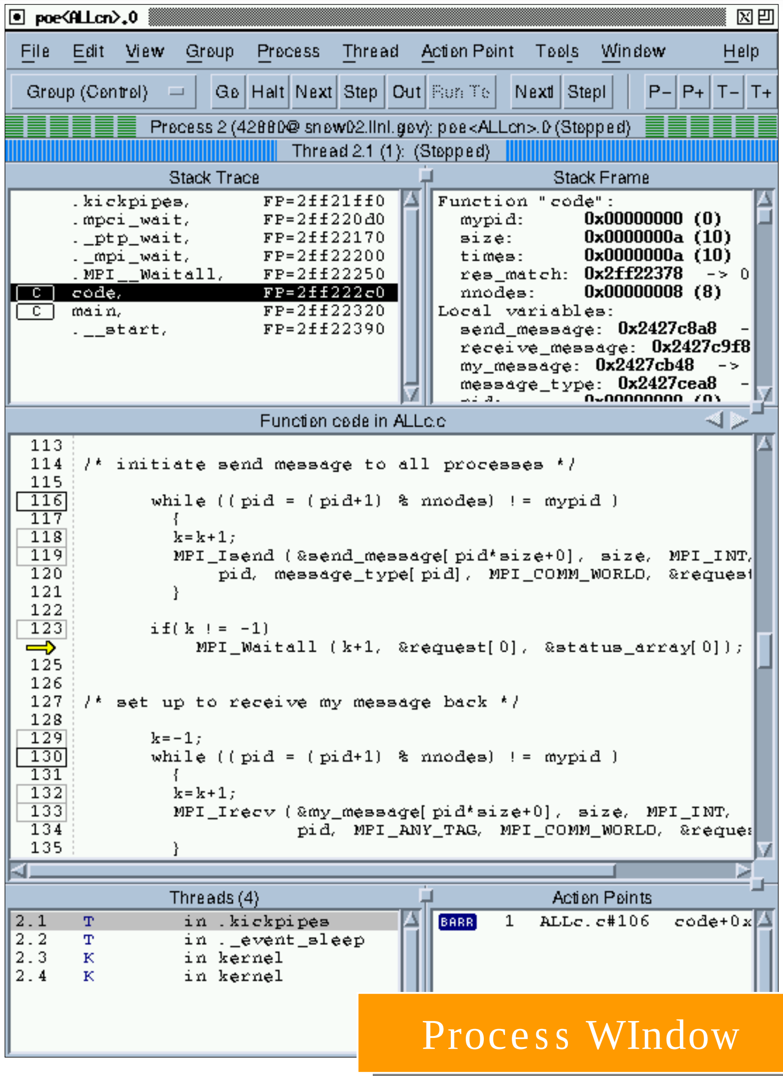

/g/g0/jdelsign/ALLc.c#main#mypid
File Edit View Tools Window Help
/g/g0/jdelsign/ALLc.c#main#mypid - 2.1

```
 (at 0x2ff22360 )  Type:  int
                 Filter:

Process              Value

poe<ALLcn>.0         0x00000000 (0)
poe<ALLcn>.1         0x00000001 (1)
poe<ALLcn>.2         0x00000002 (2)
poe<ALLcn>.3         0x00000003 (3)
poe<ALLcn>.4         0x00000004 (4)
poe<ALLcn>.5         0x00000005 (5)
poe<ALLcn>.6         0x00000006 (6)
poe<ALLcn>.7         0x00000007 (7)
```

Variable (Process Laminated)


poe<ALLcn>.0
File Edit View Group Process Thread Action Point Tools Window Help
Group (Control) Go Halt Next Step Out Run To NextI StepI P- P+ T- T+

Process 2 (42880@ snow02.llnl.gov): poe<ALLcn>.0 (Stopped)

Thread 2.1 (1): (Stopped)

Stack Trace

```
    .kickpipes,        FP=2ff21ff0
    .mpci_wait,        FP=2ff220d0
    ._ptp_wait,        FP=2ff22170
    ._mpi_wait,        FP=2ff22200
    .MPI__Waitall,     FP=2ff22250
 C  code,              FP=2ff222c0
 C  main,              FP=2ff22320
    .__start,          FP=2ff22390
```

Stack Frame

```
Function "code":
  mypid:        0x00000000 (0)
  size:         0x0000000a (10)
  times:        0x0000000a (10)
  res_match:    0x2ff22378  -> 0
  nnodes:       0x00000008 (8)
Local variables:
  send_message:  0x2427c8a8  -
  receive_message:  0x2427c9f8
  my_message:  0x2427cb48  ->
  message_type:  0x2427cea8  -
```

Function code in ALLc.c

```
/* initiate send message to all processes */

      while ((pid = (pid+1) % nnodes) != mypid )
        {
        k=k+1;
        MPI_Isend (&send_message[pid*size+0], size, MPI_INT,
            pid, message_type[pid], MPI_COMM_WORLD, &request

        }

      if(k != -1)
          MPI_Waitall (k+1, &request[0], &status_array[0]);


/* set up to receive my message back */

      k=-1;
      while ((pid = (pid+1) % nnodes) != mypid )
        {
        k=k+1;
        MPI_Irecv (&my_message[pid*size+0], size, MPI_INT,
                   pid, MPI_ANY_TAG, MPI_COMM_WORLD, &reques
        }
```

Threads (4)

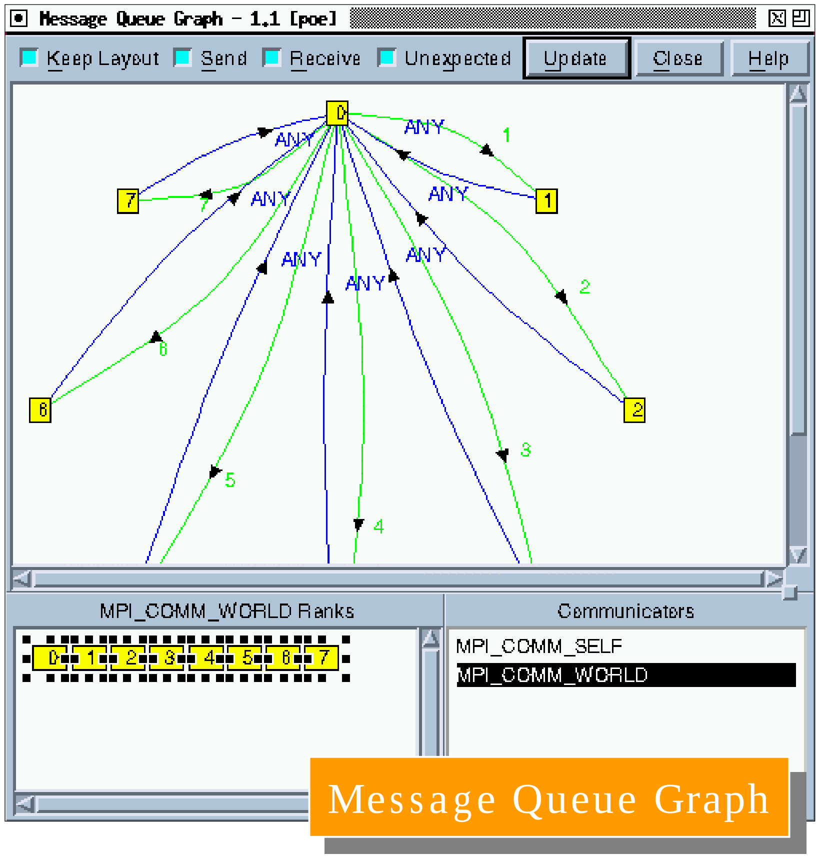
```
2.1   T      in .kickpipes
2.2   T      in ._event_sleep
2.3   K      in kernel
2.4   K      in kernel
```

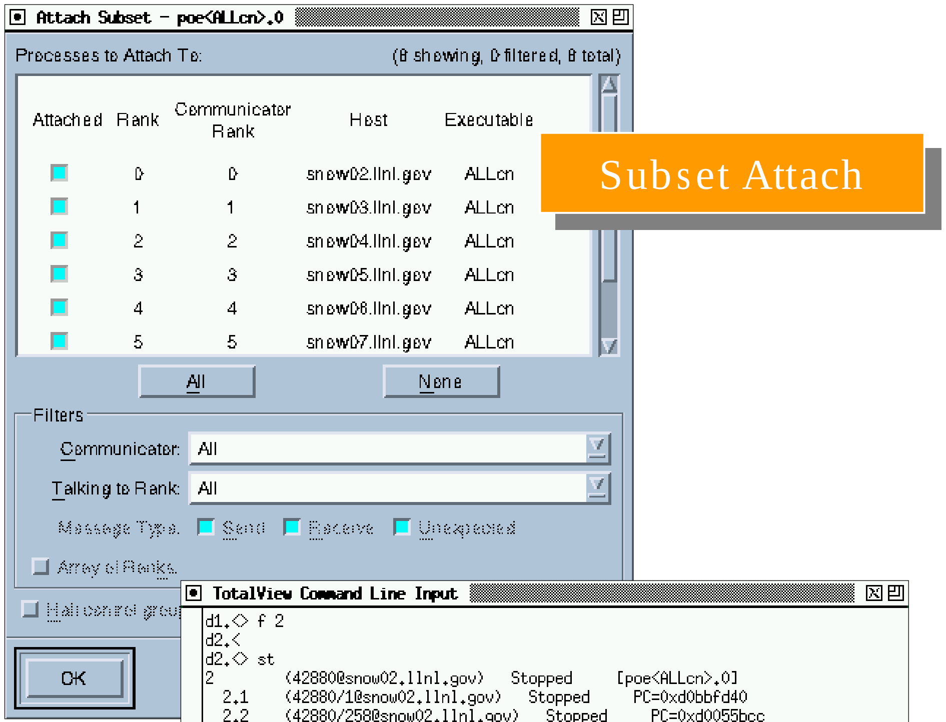
Action Points

```
BARR   1   ALLc.c#106   code+0x
```

Process WIndow

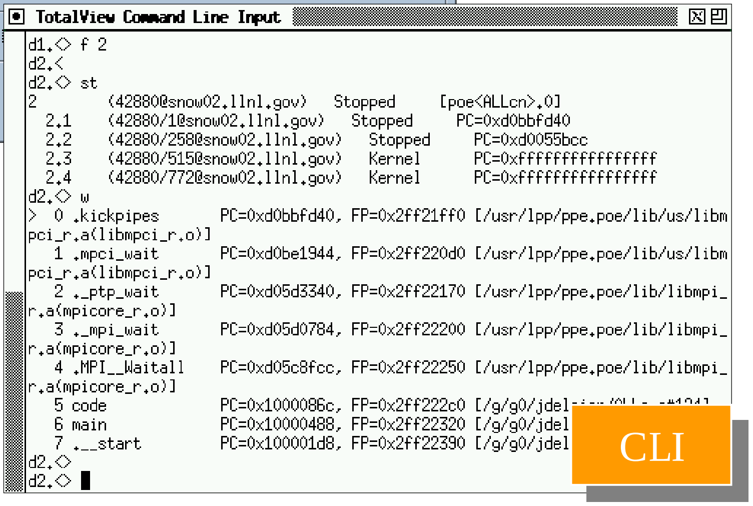
TotalView GUI & CLI (cont)





Processes to Attach To: (8 showing, 0 filtered, 8 total)

Attached	Rank	Communicator Rank	Host	Executable
🟦	0	0	snow02.llnl.gov	ALLcn
🟦	1	1	snow03.llnl.gov	ALLcn
🟦	2	2	snow04.llnl.gov	ALLcn
🟦	3	3	snow05.llnl.gov	ALLcn
🟦	4	4	snow06.llnl.gov	ALLcn
🟦	5	5	snow07.llnl.gov	ALLcn

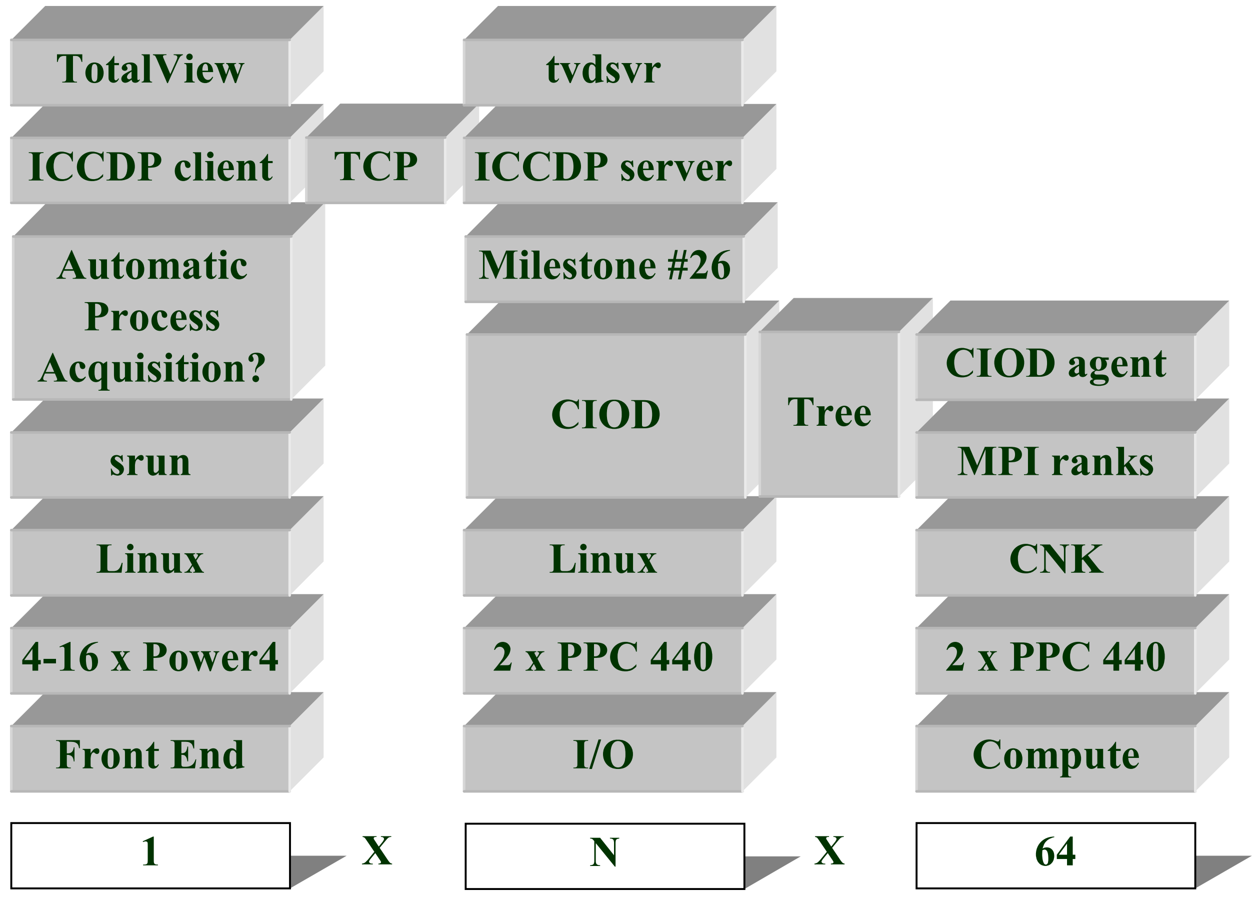


```
d1.<> f 2
d2.<
d2.<> st
2        (42880@snow02.llnl.gov)    Stopped      [poe<ALLcn>.0]
  2.1    (42880/1@snow02.llnl.gov)    Stopped      PC=0xd0bbfd40
  2.2    (42880/258@snow02.llnl.gov)    Stopped      PC=0xd0055bcc
  2.3    (42880/515@snow02.llnl.gov)    Kernel       PC=0xffffffffffffffff
  2.4    (42880/772@snow02.llnl.gov)    Kernel       PC=0xffffffffffffffff
d2.<> w
>  0 .kickpipes      PC=0xd0bbfd40, FP=0x2ff21ff0 [/usr/lpp/ppe.poe/lib/us/libm
pci_r.a(libmpci_r.o)]
   1 .mpci_wait      PC=0xd0be1944, FP=0x2ff220d0 [/usr/lpp/ppe.poe/lib/us/libm
pci_r.a(libmpci_r.o)]
   2 ._ptp_wait      PC=0xd05d3340, FP=0x2ff22170 [/usr/lpp/ppe.poe/lib/libmpi_
r.a(mpicore_r.o)]
   3 ._mpi_wait      PC=0xd05d0784, FP=0x2ff22200 [/usr/lpp/ppe.poe/lib/libmpi_
r.a(mpicore_r.o)]
   4 .MPI__Waitall   PC=0xd05c8fcc, FP=0x2ff22250 [/usr/lpp/ppe.poe/lib/libmpi_
r.a(mpicore_r.o)]
   5 code            PC=0x1000086c, FP=0x2ff222c0 [/g/g0/jdel
   6 main            PC=0x10000488, FP=0x2ff22320 [/g/g0/jdel
   7 .__start        PC=0x100001d8, FP=0x2ff22390 [/g/g0/jdel
d2.<>
d2.<> 
```

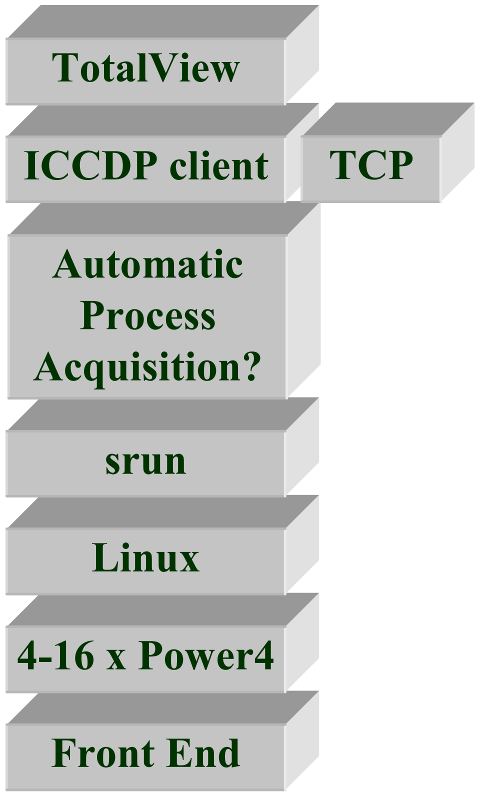
Porting TotalView to BG/L

- Etnus is porting TotalView to BG/L
- Working with IBM to
  - Iron out the details of *how* to do it
  - Collaborating on debugging interfaces
- Remainder of this talk will outline
  - General porting approach we plan to use
  - Outline of the functionality we plan to have
  - Scalability and performance ideas
- Details are subject to change
  - At participating dealers only, your mileage may vary, offer not valid in all states, some restrictions apply ☺.

BG/L TotalView Debugger Software Stack

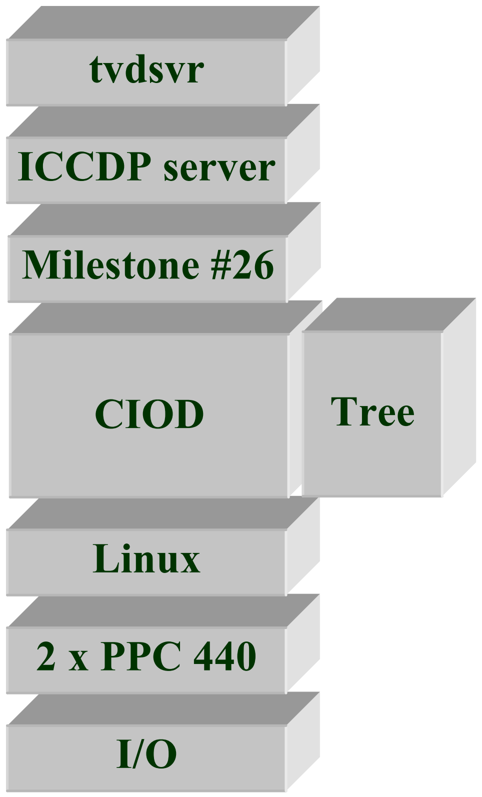


Front-End Node (FEN)



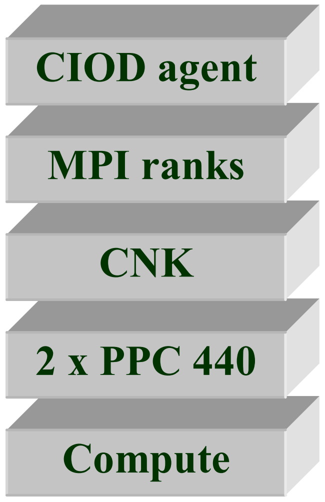
- Requires a powerful FEN
  - Server-class SMP (4-16 way)
  - Power4, 1GHz+
  - 4GB+ memory
- Runs Linux OS
- Hosts the TotalView GUI and CLI clients
- TV talks client-side ICCDP to *N* TV debugger servers over TCP/IP
- Automatic process acquisition TBD
  - May use existing approach (TV debugs mpirun)
  - May need to develop a new client/server approach (requires LLNL/IBM/Etnus collaboration)

I/O Nodes



- Reasonable horsepower for the TotalView Debugger Server (tvdsvr)
  - 700 MHz PPC 440
  - 512 MB memory
- Runs Linux OS
- Hosts the BG/L tvdsvr
- tvdsvr talks server-side ICCDP over TCP/IP to TV
- 1 server exerts trace-level control over 64 compute nodes
- Communicates with CIOD via pipe
  - To implement low-level debugging protocol
  - CIOD debug messages sent to/from CIOD agent

Compute Nodes



- Hardware
  - 2 x 700 MHz PPC 440
  - Double Hummer FPU (1 per processor)
  - 512 MB memory
- Runs CNK OS
- No part of TotalView runs here
  - So IBM knows better how this really works
- Debugging happens in CNK exception handlers (“CIOD agent” is my name)
  - Communicates with CIOD on Linux I/O node
  - Sends/receives debug messages over the tree
- TotalView will view each compute node as
  - A single process (one address space)
  - Having two threads (one thread per processor)

BG/L TotalView Features

- TotalView version 6.4 or later (mainline)
- TotalView GUI and CLI
  - Breakpoints, single-step, etc.
- Documentation
  - Electronic form, HTML help
  - Blue Gene/L specific addenda
- Languages
  - C, C++, Fortran
  - Assembler
  - Mixed languages
- Compilers
  - GCC 3
  - IBM XL / Visual Age
- MPI
  - Automatic process pickup
  - Message queue display
- Double Hummer FPU
- Data watchpoints?
  - If supported by CNK
- STL types display?
  - vector<>, list<>, map<>

Unsupported Features

- No data Visualizer
- No OpenMP
- No SHMEM, PVM
- No pthreads
- No shared libraries
- No compiled expressions
  - Interpreted expressions only
- No checkpoint restart
- No core files

Scalability and Performance

- TotalView's History of Scaling
- Defense Mechanisms for Scaling
- Scalability and Performance Philosophy
- Plans for Scaling to BG/L
- Do We Need a Paradigm Shift?

TotalView's History of Scaling

- TotalView was designed to be a parallel debugger (BBN Butterfly)
- 10 years ago, scaled to about 100 processes comfortably
- ASCI Path Forward projects
  - Developed Subset Attach feature
  - Scalability was increased to
    - Handle about 1,000 processes comfortably
    - Handle about 2,000 processes less comfortably
  - But problems start at about 3,000 processes

Defense Mechanisms for Scaling

- Debug a smaller job
  - Typically debugging 1 process!
  - Debugging 4 to 32 processes is very common
  - Some routinely debug 100 processes
  - Rarely 1,000 processes or more
- Developed Subset Attach feature
  - Debug a subset of processes in a large parallel job
  - On job launch or attach
  - Fan out attach during a session
    - Based on MPI process communication state
    - Data values in a scalar array

Scalability and Performance Philosophy

- Must scale in three dimensions
  - Resource consumption: Do we fit within system limits?
  - Runtime performance: Are we responsive to the user?
  - Presentation of information: Can we present information in an easily digested format?
- Must actively work on scalability and performance
  - Machines and programs are getting bigger
  - Feature additions tend to slow things down
  - Continuously test and measure, using a hands-on approach, which requires access to the
    - End-user's machine resources
    - End-user's application
    - End-user's usage scenarios

ETNUS TOTALVIEW

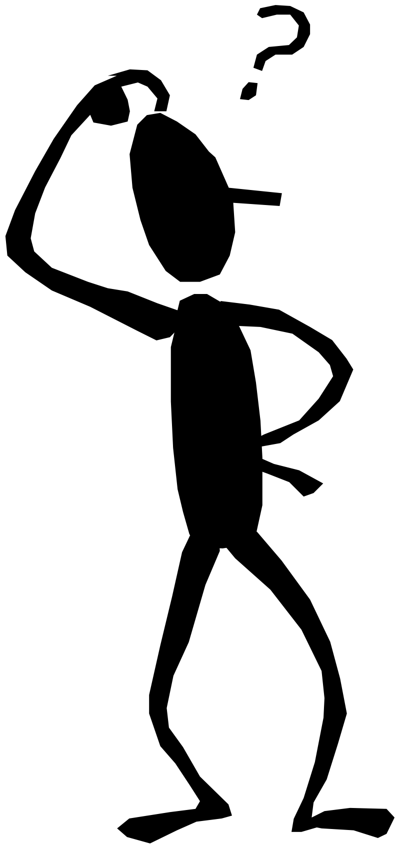
Plans for Scaling to BG/L

- Special effort will be required for BG/L scale machines
- Performance and scalability approaches under consideration
  - Restructured finite-state machine
  - Push more computation into the tvdsvr
  - Restructure messaging: Async TV⇔tvdsvr, “psychic”, aggregated, optimistic
  - Multi-threading TV and/or tvdsvr
- Haven’t thought too hard about how to scale the GUI yet

Do We Need a Paradigm Shift?

- Above concerned with scaling TV, but I have a few questions for you …
- Do users really want to debug 64,000 processes?
- What goes wrong at 64,000 vs. 1,000?
  - Correctness? Performance? Something else?
- Do we need additional facilities?
  - Lightweight "watchdog" debugger
  - High-scale lightweight event tracer
  - Hardware performance counters & tools

Questions and Answers



- www.etnus.com
- info@etnus.com