Cray Tools Current Status and Future Plans

Luiz DeRose
Tools Manager
Cray Inc.
ldr@cray.com

Outline

- **Current status**
  - **Debugger**
  - **Performance Analysis Toolkit**
- **Future directions**
- **User feedback**

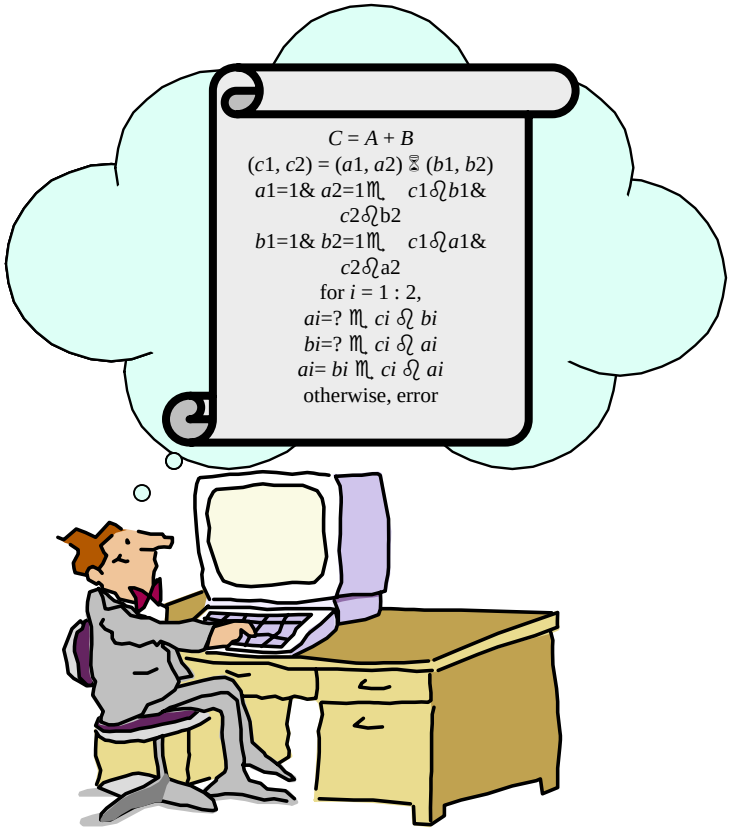
CUG 2004

Cray TotalView Debugger

- **Version 6.3.1.0 – Shipped 05/01/04**
  - **Feature highlights**
    - **CLI and GUI variants**
    - **Core file support (including 64 bit files)**
    - **PThread support for core files**
    - **Support for MSP and SSP**
    - **Viewing info for any SSP of an MSP**
  - **New features in 6.3.1**
    - **Support “aprun” launching of jobs from within the debugger**
    - **Support access of information from SSPs 1-3**
  - **New features planned for next release**
    - **Live PThread**
    - **CAF/UPC support**
    - **Support for vector registers**

CUG 2004

Performance Tools Challenge



- **User's mental model of program does not match with executed version**
  - **Users need to understand how software and hardware resources are utilized**
  - **Performance tools must be able to revert this semantic gap**

CUG 2004

Cray PAT

- **Performance analysis toolkit**
  - **Applied to applications for single or multiple PEs**
    - **Shared memory**
    - **Distributed memory**
    - **MSP mode**
    - **SSP mode**
  - **Collect data at all levels of parallelism**
    - **SSP**
    - **Process**
    - **Thread**
  - **Information**
    - **Call stack**
    - **Thread**
    - **Hardware performance counters**

CUG 2004

Cray PAT Components

- pat_hwpc
  - Stand-alone utility that executes an application, collect specified HW performance counter events, and writes a summary report
- pat_build
  - Utility that instruments the application
- Cray PAT run-time library
  - Transparent to the user
  - Collects performance data during execution
  - Writes data file
- pat_report
  - Utility to create a performance report
- pat_help
  - Help utility

CUG 2004

PAT_HWPC

- Collects hardware performance counters information for an application
- No instrumentation required
  - pat_hwpc [options] a.out
  - Accepts various hardware counters groups
- Results in report with raw counts and derived metrics for the whole execution
  - Hardware counters summed across threads per process

PAT_HWPC Output

```
Command executed:  aprun -n1 ./swim
Exit status        0
Host name & type   water crayx1 400 MHz
Operating system   UNICOS/mp 2.4.18 05160857
Text page size     16 Mbytes
Other page size    16 Mbytes
Start time         Thu May 20 08:52:43 2004
End time           Thu May 20 08:52:48 2004

Elapsed time       5.653 seconds
User time          3.569 seconds   63%
System time        1.596 seconds   28%

Resource usage by process and module:
PE     PID  User time  System time  Module  Memory resident
            (seconds)    (seconds)            size (MBytes)

 0  623172   3.561109     1.549820      28       160.125000

Accounting data by process:
     Read &   Read &   Block      Raw       Run Q    Num.
      Write    Write  I/O Wait  I/O Wait     Wait     of
PE  (MBytes)   Calls   (secs)    (secs)     (secs)  Swaps

 0  0.836510       7  0.001527  0.000000  0.042894      0
    0.001341      19
```

Processor Metrics

```
P counter data
CPU Seconds                              3.503683 sec
Cycles                 1559.895M/sec   5465376812 cycles
Instructions graduated  352.665M/sec   1235625287 instr
Branches & Jumps          4.694M/sec     16445103 instr
Branches mispredicted     0.517M/sec      1812301 misses    11.020%
Correctly predicted       4.176M/sec     14632802 misses    88.980%
Vector instructions     133.564M/sec    467965318 instr     37.873%
Scalar instructions     219.101M/sec    767659969 instr     62.127%
Vector ops             8546.749M/sec  29945097182 ops
Vector FP adds         2919.849M/sec  10230224517 ops
Vector FP multiplies   1873.126M/sec   6562837581 ops
Vector FP divides etc    75.346M/sec    263987335 ops
Vector FP misc            0.991M/sec      3471668 ops
Vector FP ops          4869.311M/sec  17060521101 ops     100.000%
Scalar FP ops             0.016M/sec        56894 ops       0.000%
Total  FP ops          4869.327M/sec  17060577995 ops
FP ops per load                             1.325 flops/load
Scalar integer ops        5.018M/sec     17580836 ops
Scalar memory refs        1.351M/sec      4733496 refs      0.037%
Vector TLB misses         0.000M/sec         1358 misses
Scalar TLB misses         0.000M/sec          939 misses
Instr  TLB misses         0.000M/sec          338 misses
Total  TLB misses         0.001M/sec         2635 misses
Dcache references         1.327M/sec      4651006 refs     98.257%
Dcache bypass refs        0.024M/sec        82490 refs      1.743%
Dcache misses             0.365M/sec      1280046 misses
```

CUG 2004

Processor (Vector) Metrics

Vector integer adds	0.227M/sec	793732	ops	
Vector logical ops	0.612M/sec	2144235	ops	
Vector shifts	0.452M/sec	1583946	ops	
Vector int ops	1.291M/sec	4521913	ops	
Vector loads	2697.276M/sec	9450398784	refs	
Vector stores	976.896M/sec	3422733702	refs	
Vector memory refs	3674.172M/sec	12873132486	refs	**99.963%**
Scalar memory refs	1.351M/sec	4733496	refs	0.037%
Total memory refs	3675.523M/sec	12877865982	refs	
Average vector length		**63.990**		
A-reg Instr	200.926M/sec	703980416	instr	
Scalar FP Instr	0.016M/sec	56894	instr	
Syncs Instr	3.536M/sec	12388143	instr	
Stall VLSU	11.492secs	4596771918	clks	
Stall VU	12.253secs	4901069010	clks	
Vector Load Alloc	2697.025M/sec	9449519404	refs	
Vector Load Index		0	refs	
Vector Load Stride	1.898M/sec	6651312	refs	
Vector Store Alloc	976.529M/sec	3421447727	refs	
Vector Store Stride	1.898M/sec	6650880	refs	

CUG 2004

Cache & Memory Metrics

E counter data		
Alloc requests	519.941M/sec	1821708706
Dcache Inval Events	0.146M/sec	509828
Evictions	247.044M/sec	865563731
Forwarded	0.061M/sec	212489
FwdGet	0.023M/sec	79829
FwdReadAll	0.005M/sec	15993
FwdReadShared		0
Inval	0.001M/sec	5132
Local Inval	0.108M/sec	376886
Misses	247.085M/sec	865707197
Nacks	0.000M/sec	2
Notifies	122.907M/sec	430627872
Requests	526.605M/sec	1845055615
Update		0
Update Nack		0
WriteBacks	123.843M/sec	433907781

M counter data		
Forwarded	0.138M/sec	484615
In Remote	0.075M/sec	263732
Inval	0.002M/sec	7429
Nacks		0
Noncached	301.041M/sec	1054752202
PendDrop		0
Requests	436.178M/sec	1528229924
Shared		0
SupplyDirtyInv	0.024M/sec	83680
SupplyDirtySh	0.000M/sec	14
SupplyExcl	0.027M/sec	95583
SupplyInv	0.085M/sec	296706
SupplySh	0.002M/sec	8627
Trans Buffer lb hit	0.037M/sec	127896
Update		0
UpdateNack	0.000M/sec	3

CUG 2004

PAT_BUILD

- Supports
  - Fortran, C, C++, CAF, UPC
  - MPI, SHMEM, OpenMP, pThreads
- Performance measurement
  - Asynchronous (interrupt-based)
  - Synchronous (trace-based)
    - Predefined function groups
- Hardware counters events from three chips
  - “P” Processor
  - “E” Cache
  - “M” Memory
- Source code mapping
  - Call stack
  - Line numbers
- API for fine grain instrumentation

CUG 2004

PAT_BUILD

- **No recompilation required**
  - **`pat_build` instruments an executable program with a single link**
    - **By default, the program is instrumented for an asynchronous experiment**
- **Execute instrumented program just as the original**
- **Environment variables allow control over various run-time features**
  - **Experiment, rate, call stack, …**
  - **No need to re-instrument the program to effect different data collection**
- **Creates a binary experiment data file**

CUG 2004

Evaluating the Results

- pat_report analyzes data and displays a formatted textual report
  - Export data to XML or spreadsheet formats
  - Control over how and what data is displayed
  - Control over appearance of report
- Time spent at function, block, source line
- Call trees (caller-callee relationships)
- Per-process, per-thread, per-SSP granularity
- Hardware Performance counters event values

CUG 2004

Cray PAT Future Directions

- User interface to simplify Cray PAT usage
  - New pat_run scripts for common requests (easier to remember)
    - pat_run –O mflops,vl a.out+pat
    - Meaningful way for selection of hardware counters
- More complete OpenMP and thread support
  - Information on
    - Load balance
    - Time on barriers
    - Run-time library overhead
  - Adoption of the proposed POMP Interface
- Run-time data aggregation
  - Reduce experiment data file size
  - Event profile (synchronous)
    - Equivalent to pat_hwpc report at function level
- Enhance and optimize report analysis

CUG 2004

Collaborations

- **Interactions with lead researchers in the performance measurement and analysis field**
  - **Jack Dongarra's group at UTK**
    - **PAPI port to all Cray systems**
  - **Bart Miller's group at U. Wisconsin**
    - **Dyninst port to X1, XD1, and Red Storm**
  - **Allen Malony's group at U. Oregon**
    - **TAU infrastructure on X1, XD1, and RS**
  - **Bernd Mohr at Forschungszentrum Jülich**
    - **KOJAK expert system being ported to X1**
  - **Jeff Vetter at ORNL**
    - **MPI Profiler port to X1 and RS**
  - **European Union APART working group**
    - **Automatic Performance Analysis: Resources and Tools**

CUG 2004

Open Research Problems

- **Scalability**
  - **Tools support for large parallel systems**
  - **Support long running programs**
  - **Distributed data analysis**

Red Storm

CUG 2004

Conclusions

- **Guidelines for tools builder**
  - **Don’t re-invent the wheel**
    - **Tools development is not simple**
  - **Support Fortran (as well as C, C++, CAF & UPC)**
  - **Make it simple**
    - **Users don’t like to use complex tools**
    - **If one cannot learn to use it quickly it won’t be used**
  - **Make it thread-safe!!**
  - **Make it scalable!!**

CUG 2004