***S**calable **P**erformance **M**onitor: an Integrated Tool for Dynamically Evaluating Application Performance*

Federico Bassetti, Jeff Brown, Phyllis Gustafson, Jack Horner, and MaryDell Tholburn

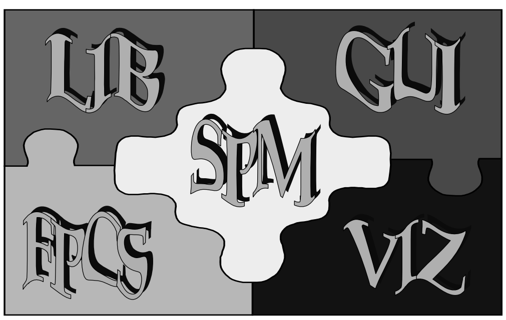
in collaboration with
George Ho, and **Phil Mucci**
University of Tennessee, Knoxville

High Performance Computing Environments Group
Los Alamos National Laboratory
http://www.lanl.gov/orgs/cic/cic8/

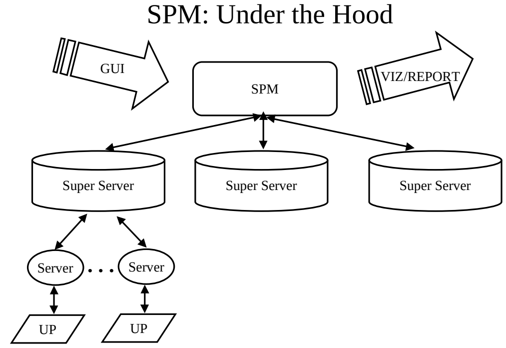
Objectives

- Developing performance tools with a focus on scalability and ASCI programming models
- Focus on memory effects: the MUTT project
- Tool infrastructure to allow dynamic extraction and analysis of performance data: Scalable Performance Monitor (SPM)
- Plans to develop/integrate messaging and cache analysis capability

SPM Architecture



SPM: Under the Hood



Free **P**robe **C**lass **S**erver

- **Immediate Goal**: To *dynamically* generate physical memory layout data from a running parallel program on the O2K.
- **Secondary Goal**: To provide a free, slim, portable and lightweight framework to dynamically instrument parallel applications with arbitrary probe libraries.

FPCS Implementation

A bottom up design…

- MUTT library
- DynInst functionality
- Client/Server architecture
- Interactive control
- Future uses

FPCS Future

- SPM integration at LANL
- Open-Source Profiling for IBM ACTC
  - PAPI metrics
  - Statistical line-by-line
  - Subroutine level
- Net accessible CVS Repository at UTK

Paradyn Week 2000 | Ptools Team -- ptools_team@lanl.gov | Los Alamos | 7

MUTT Probe Design

- Compile MUTT as a shared library
- Make MUTT probe shared library
  - `MUTT_start()`
    - Open data file
    - Set up interval timer to call MUTT_probe()
  - `MUTT_probe()`
    - Call MUTT's `cm_process_dist()` to dump the entire process' address space
    - Write to data file
  - `MUTT_stop()`
    - Disable interval timer
    - Flush/Close data file

Paradyn Week 2000 | Ptools Team -- ptools_team@lanl.gov | Los Alamos | 8

DynInst Functionality

- No need for the abstract syntax language.
  - Probes are easily tested when in separate libraries.
- Use one-time probe execution to call `MUTT_start()` and `MUTT_stop()`.
- Use process control features
- Client manages handles, pointers to DynInst data structures in the Server

Paradyn Week 2000 | Ptools Team -- ptools_team@lanl.gov | Los Alamos | 9

Client/Server Architecture

- Well defined, fixed length, request/reply data structures in header file.
- Synchronous communication for simplicity
- 1 Client issues commands to…
- 1 Super server per box relays commands to..
- 1 Server per PID which receives commands, executes them and returns a response.

Paradyn Week 2000 | Ptools Team -- ptools_team@lanl.gov | Los Alamos | 10

Memory Utilization Tracking Tool: the *MUTT*

- User-callable library gives process snapshot
- Reports size and location of memory regions
- Estimates latency for remote memory access
- Tracks program accesses
- Ensight visualization of MUTT data
- Results validated against ref count and dlook data

Paradyn Week 2000 | Ptools Team -- ptools_team@lanl.gov | Los Alamos | 11

MUTT Mechanics

Pages = mutt_process_dist(data_destination, identifier)
    Destination: screen, file, viz (muttviz invoked)
    Identifier: User-supplied tag (e.g., MPI rank or PID)

MUTT output:
machine-id, pid, process-node, region memory-node, latency pages accesses

muttviz:
    Generates Ensight files to map mutt data onto a 3d representation of a 128-processor O2K

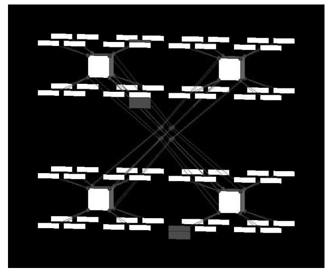
Other user-written analysis programs have been developed

Paradyn Week 2000 | Ptools Team -- ptools_team@lanl.gov | Los Alamos | 12

Access Profiling

- Statistical profiling via dprof-like mechanism
  - Register a handler for SIGPROF
  - Start triggering mechanism
    - Interval timer or secondary cache miss overflow
- Upon receipt of a SIGPROF the handler:
  - Scans the instruction stream from current PC location for next load/store instruction
  - Parses out virtual address
  - Logs access
- Mutt calls libmuttprof routine if profiling enabled to return accesses for virtual address and reports out for corresponding physical address

2-D Heat Conduction Test Problem

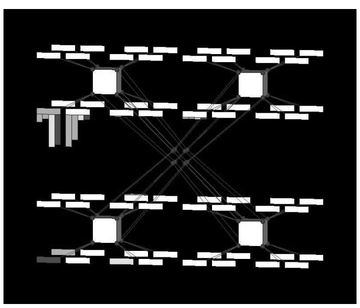


Single process

Some memory local, but a large portion at remote location.

Irix 6.5.6 (open)

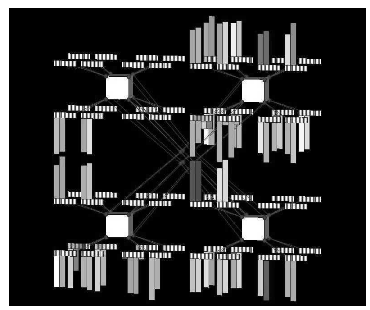
4 Process RAGE Run



Almost no locality for heap, stack or text

Irix 6.5.6 (open)

64 process RAGE Run

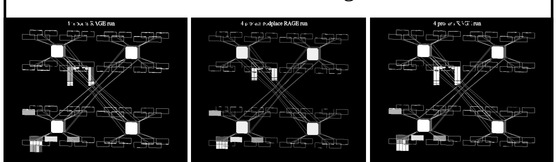


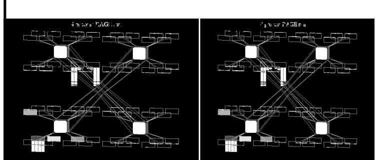
Nodes are over-subscribed

Minimal memory Locality

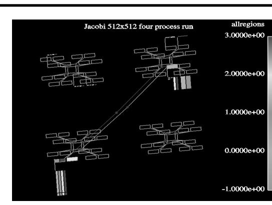
Irix 6.5.6 (open)

4 Process RAGE run with region breakdowns

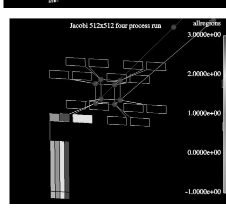


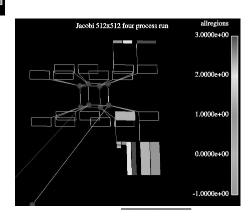


MPI buffers required special MPI routines provided by SGI (Howard Pritchard)

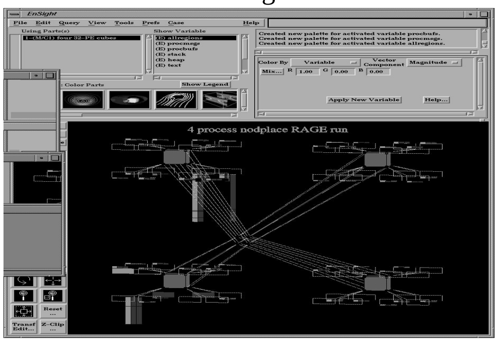


Note:
• accesses captured for data segments only
• most accesses are non-local
•*no* local accesses for green and red processes





MUTTviz Ensight Interface



Current Directions

- Thread support (e.g. OpenMP programs):
  - thread-safe mutt,
  - thread-stack awareness,
  - associate heap pages with thread that allocated it
- Investigate ways to measure router contention
- Dynamic instrumentation with MUTT probes
- Viz extensions
  - real-time (Ensight reader)
  - Java3D for open source release

Current Directions (more)

- Automate optimal process and page placement
  - Local memory pages NUMA-close to process/thread
  - Co-locate processes with shared memory dependencies (dprof/dplace mechanism designed for threads)
  - Isolate multiple concurrent MPI jobs
  - LSF integration (SMP topology awareness)
- Work external release (Ptools consortium project)
- Apply to LANL codes/frameworks
- Move on to study cache and message effects

SPM: Command Line Interface

- Process Control
  - attach, detach
  - run, stop, end
  - breakpoint
- Instrumentation
  - insert
  - replace
  - remove

SPM Capabilities

	cache	numa	hw pc	msg trc
sample		X	X	X
snap		X	X	
trace	X	X	X	X
viz	X	X	X	X
report	X	X	X	X

SPM: Example

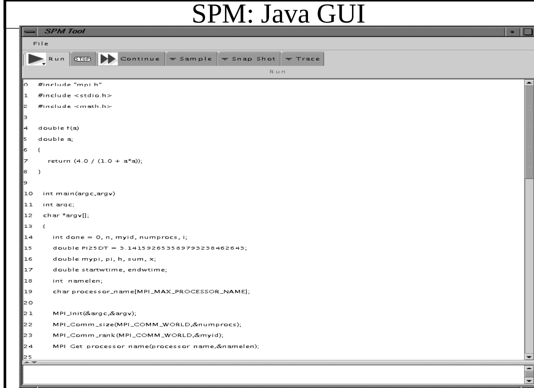
>mpirun *-np* p *-m* m1,m2 test	Run mpi job on multiple processors and across boxes
>spm -a test	SPM gathers pid associated with test and attaches a server to each parallel process (superserver>>>server>>>dyninst
sample numa	Dynamically load the mutt library, (via FPCS) set up
run	Restart job that now has instrumentation
viz numa	Process and visualize data collected by mutt
quit	Detach all servers and remove all instrumentation calls, job's execution continues

SPM Java GUI

- Source code display and navigation
- Insert/remove various forms of instrumentation
- Process control, stop job at a point of interest
- Display performance data in various way
- Designed to allow other software as “plug in”

Paradyn Week 2000 Ptools Team -- ptools_team@lanl.gov 25

SPM: Java GUI



Paradyn Week 2000 Ptools Team -- ptools_team@lanl.gov 26

Conclusions

- SPM capabilities, complete matrix
- process control to support grouping
- scalability (30T)
- portability
  - java gui
  - java viz
- plug-in of other software
  - mutt
  - vampire

Paradyn Week 2000 Ptools Team -- ptools_team@lanl.gov 27