Parallelism V
HPC Profiling

John Cavazos

Dept of Computer & Information Sciences

University of Delaware

Lecture Overview

- Performance Counters
- Profiling
  - PAPI
  - TAU
  - HPCToolkit
  - PerfExpert

Performance Counters

- Special purpose registers built into the processor microarchitecture
  - Monitor hardware-related activity within the computer
- Useful for performance analysis and tuning
  - Provide low-level information that can not be obtained with software profilers

source: http://pcsostres.ac.upc.edu/eitm/lib/exe/fetch.php/slides:pmc.pdf

Performance Counters

- You can get insight into…
  - Whole program timing
  - Cache behaviors
  - Branch behaviors
  - Memory and resource access patterns

source: http://tau.uoregon.edu/tau.ppt

Performance Counters

- You can get insight into…
  - Pipeline stalls
  - Floating point efficiency
  - Instructions per cycle …

Identify Program Bottlenecks

source: http://tau.uoregon.edu/tau.ppt

PAPI

- **P**erformance **A**pplication **P**rogramming **I**nterface
- To design, standardize and implement a portable and efficient API
- C and Fortran Interface

source: http://pcsostres.ac.upc.edu/eitm/lib/exe/fetch.php/slides:pmc.pdf

PAPI Utility Commands

- papi_avail: Standard preset over 100 events

```
epark@aji:~$ papi_avail
Available events and hardware information.
--------------------------------------------------------------------------------
PAPI Version             : 4.1.1.0
Vendor string and code   : GenuineIntel (1)
Model string and code    : Intel(R) Core(TM)2 Quad CPU    Q9650  @ 3.00GHz (23)
CPU Revision             : 10.000000
CPUID Info               : Family: 6  Model: 23  Stepping: 10
CPU Megahertz            : 2003.000000
CPU Clock Megahertz      : 2003
Hdw Threads per core     : 1
Cores per Socket         : 4
NUMA Nodes               : 1
CPU's per Node           : 4
Total CPU's              : 4
Number Hardware Counters : 5
Max Multiplex Counters   : 512
--------------------------------------------------------------------------------
The following correspond to fields in the PAPI_event_info_t structure.

    Name        Code    Avail Deriv Description (Note)
PAPI_L1_DCM  0x80000000  Yes   No   Level 1 data cache misses
PAPI_L1_ICM  0x80000001  Yes   No   Level 1 instruction cache misses
PAPI_L2_DCM  0x80000002  Yes   Yes  Level 2 data cache misses
PAPI_L2_ICM  0x80000003  Yes   No   Level 2 instruction cache misses
PAPI_L3_DCM  0x80000004  No    No   Level 3 data cache misses
```

PAPI Utility Commands

- papi_native_avail: Any event countable on the specific

```
epark@aji:~$ papi_native_avail
Available native events and hardware information.
--------------------------------------------------------------------------------
PAPI Version             : 4.1.1.0
Vendor string and code   : GenuineIntel (1)
Model string and code    : Intel(R) Core(TM)2 Quad CPU    Q9650  @ 3.00GHz (23)
CPU Revision             : 10.000000
CPUID Info               : Family: 6  Model: 23  Stepping: 10
CPU Megahertz            : 2003.000000
CPU Clock Megahertz      : 2003
Hdw Threads per core     : 1
Cores per Socket         : 4
NUMA Nodes               : 1
CPU's per Node           : 4
Total CPU's              : 4
Number Hardware Counters : 5
Max Multiplex Counters   : 512
--------------------------------------------------------------------------------
The following correspond to fields in the PAPI_event_info_t structure.

Event Code   Symbol  | Long Description |
--------------------------------------------------------------------------------
0x40000000   UNHALTED_CORE_CYCLES  | count core clock cycles whenever the cloc |
            | k signal on the specific core is running (not halted). Alias to e |
            | vent CPU_CLK_UNHALTED:CORE_P                                      |
--------------------------------------------------------------------------------
```

PAPI Utility Commands

- papi_event_chooser
  - Whether a given event is available NATIVE or PRESET
  - Whether given events can be used together
  - > papi_event_chooser PRESET PAPI_L2_TCH

```
    Name         Code      Deriv Description (Note)
PAPI_L2_TCM  0x80000007  No    Level 2 cache misses
PAPI_TOT_INS 0x80000032  No    Instructions completed
PAPI_TOT_CYC 0x8000003b  No    Total cycles
PAPI_L2_TCA  0x80000059  No    Level 2 total cache accesses
-------------------------------------------------------------------------
Total events reported: 4
event_chooser.c                                    PASSED
```

PAPI High Level Functions

- `PAPI_num_counters(void)`: Return the number of h/c available on the system
- `PAPI_library_init(int version)`: Initialize the PAPI library
  - version: Using `PAPI_VER_CURRENT`
- `PAPI_is_initialized()`: Return the initialized state of the PAPI library

source: http://icl.cs.utk.edu/projects/papi/wiki/

PAPI High Level Functions

- `PAPI_create_eventset(int *EventSet)`: Create a new empty PAPI event set
  - EventSet is itialized by `PAPI_NULL`
- `PAPI_add_event(int EventSet, int EventCode)`: Add single event
- `PAPI_add_events(int EventSet, int *EventCode)`: Add multiple events
- `PAPI_remove_event(int EventSet, int EventCode)` or `PAPI_add_events(int EventSet, int *EventCode)`: Remove event(s)

source: http://icl.cs.utk.edu/projects/papi/wiki/

PAPI High Level Functions

- `PAPI_start(int EventSet)`: Start couting hardware events in an EventSet
- `PAPI_read (int EventSet, long_long * values)`: Read hardware counters from an EventSet
- `PAPI_stop(int EventSet, long_long * values)`: Stop counting harware events in an EventSet
- `PAPI_shutdown()`: finish using PAPI and free all related resources

source: http://icl.cs.utk.edu/projects/papi/wiki/

Using PAPI

- Example of PAPI usage will show up here
  - `#include<papi.h>` in matmul-seq-papi.c
  - Add initialization functions for PAPI library, and Add `PAPI_start` before the code you want to measure, after `PAPI_read`, then `PAPI_stop`.
  - `> gcc -O2 -g -o matmul-seq-papi matmul-seq-papi.c -lpapi`
  - `> ./matmul-seq-papi`

Example of PAPI

- In matmul-seq-papi.c
  - Need to specify `int eventCode[] = {PAPI_TOT_CYC, PAPI_L1_DCM, PAPI_L1_ICM, PAPI_L2_DCM, PAPI_L2_ICM}`
- Run `./matmul-seq-papi`

```
11739215199, 1081351098, 747,
384594, 29
```

Example of PAPI

- Demo

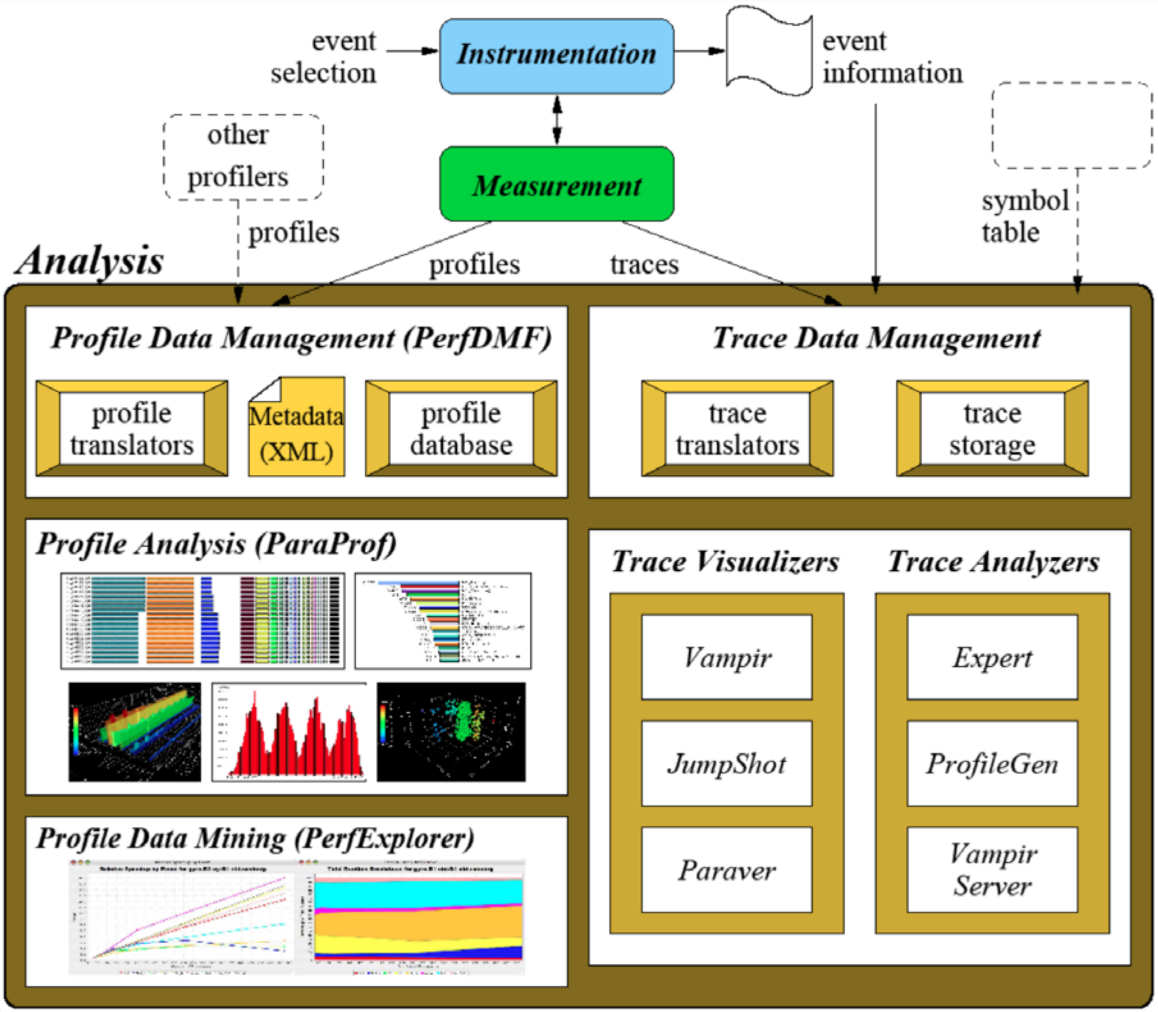
TAU

- TAU (Tuning and Analysis Utilities)
  - http://tau.uoregon.edu
  - Program and performance analysis tool framework for (parallel) programs
  - Automatic instrumentation for functions, loops, and so on
  - Static and dynamic analysis of programs written in C, C++, Fortran, Python, and Java.

source: http://tau.uoregon.edu/tau.ppt

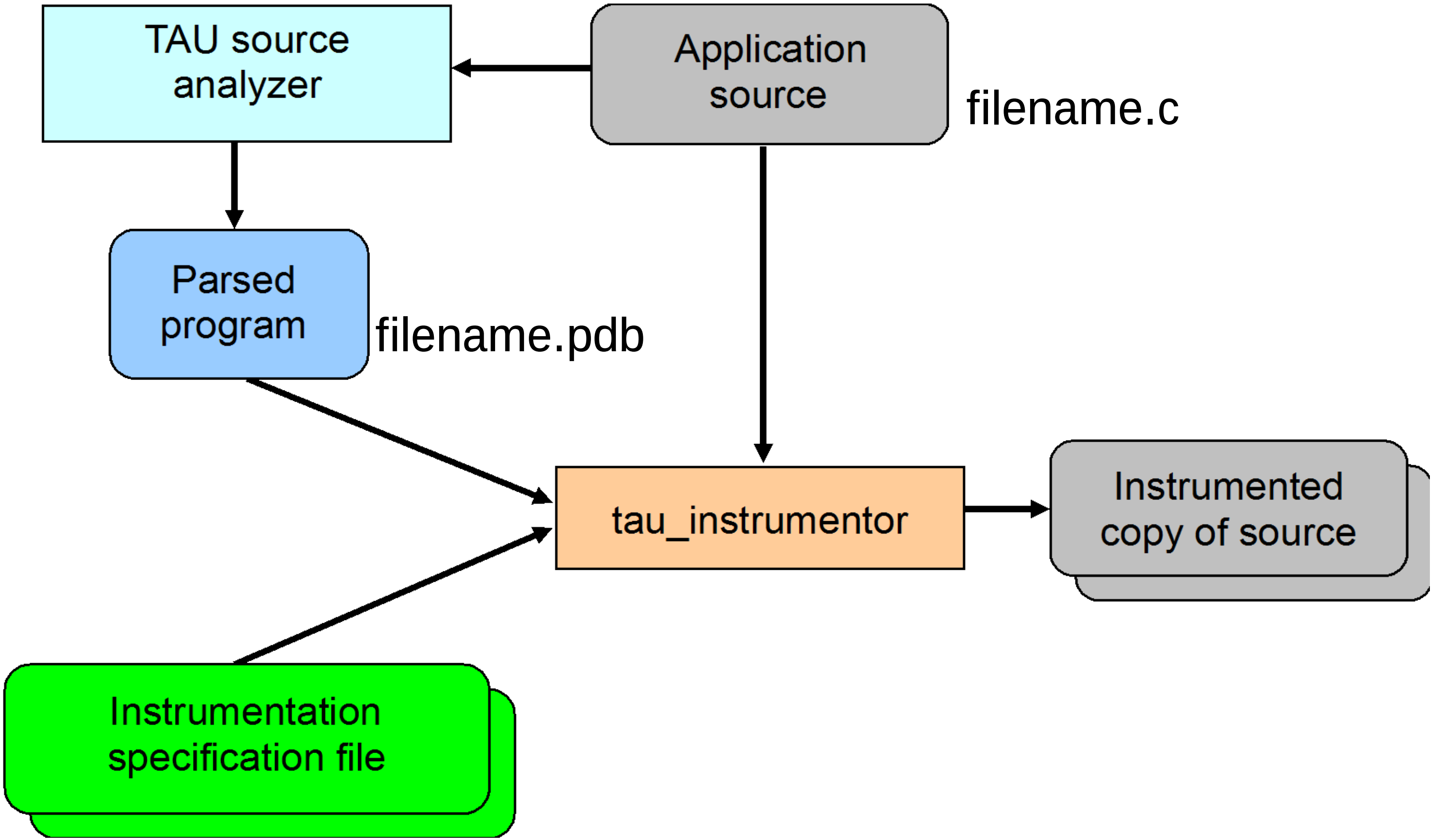
TAU Performance System

- Instrumentation
- Measurement
- Analysis
- Visualization



source: http://tau.uoregon.edu/tau.ppt

Automatic Instrumentation



source: http://tau.uoregon.edu/tau.ppt

Measurement and Analysis

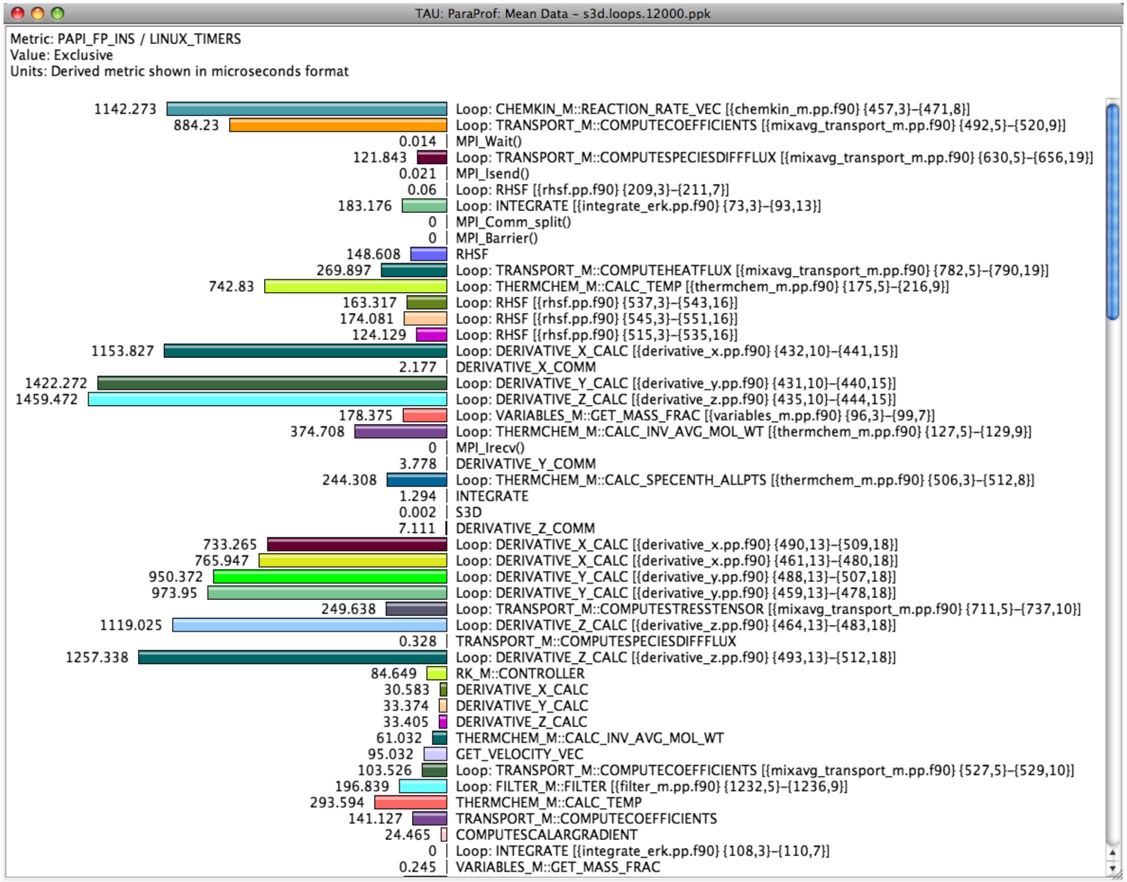
- Compile and run instrumented code
- Define metrics you want to measure
  - `TAU_METRICS=TIME:PAPI_FP_INS:…`
- You will see directories named with metric name
  - `cd MULTI__GET_TIME_OF_DAY`
  - `pprof`

```
Reading Profile files in profile.*

NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        8,003        8,003           1           0    8003555 int main() C
```

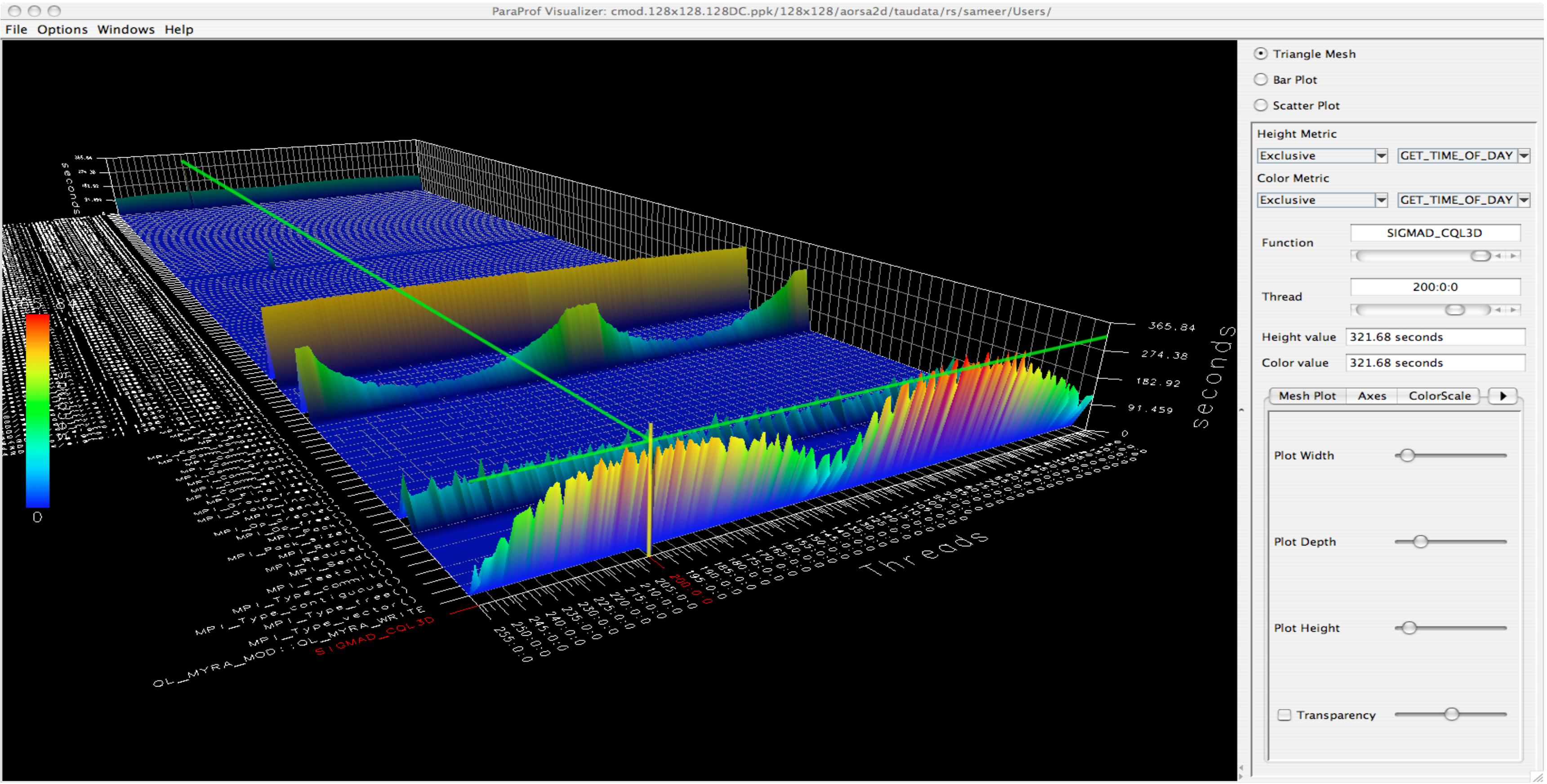
Visualization

- Using paraprof, identify bottleneck



source: http://tau.uoregon.edu/tau.ppt

Using 3D paraprof Browser



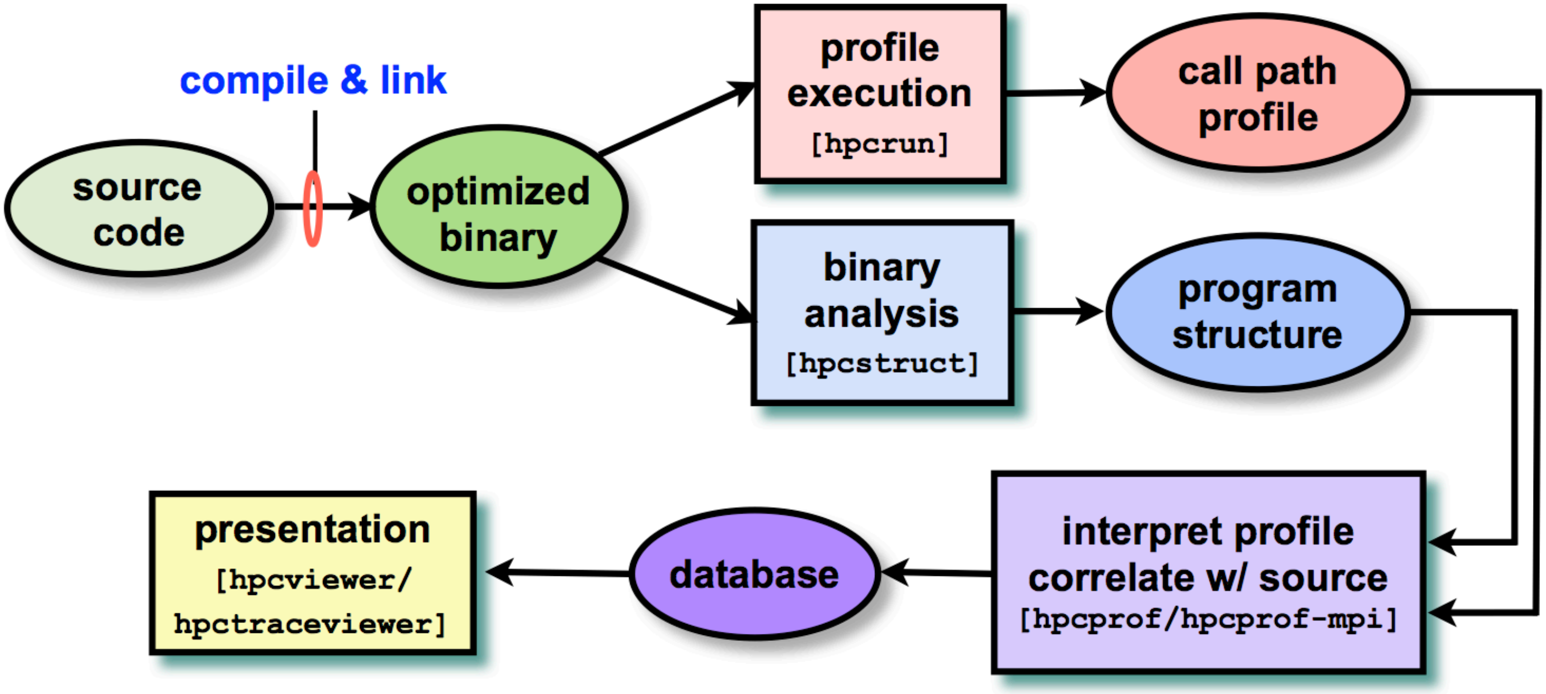
source: http://tau.uoregon.edu/tau.ppt

HPCToolkit

- http://hpctoolkit.org
- Measurement and analysis of program performance
- Using statistical sampling of timers and hardware performance counters
- Platforms supported: Linux X86_64, Linux-x86, Linux-Power, Cray XT/XE/XK, IBM Blue Gene/Q, Blue Gene/P

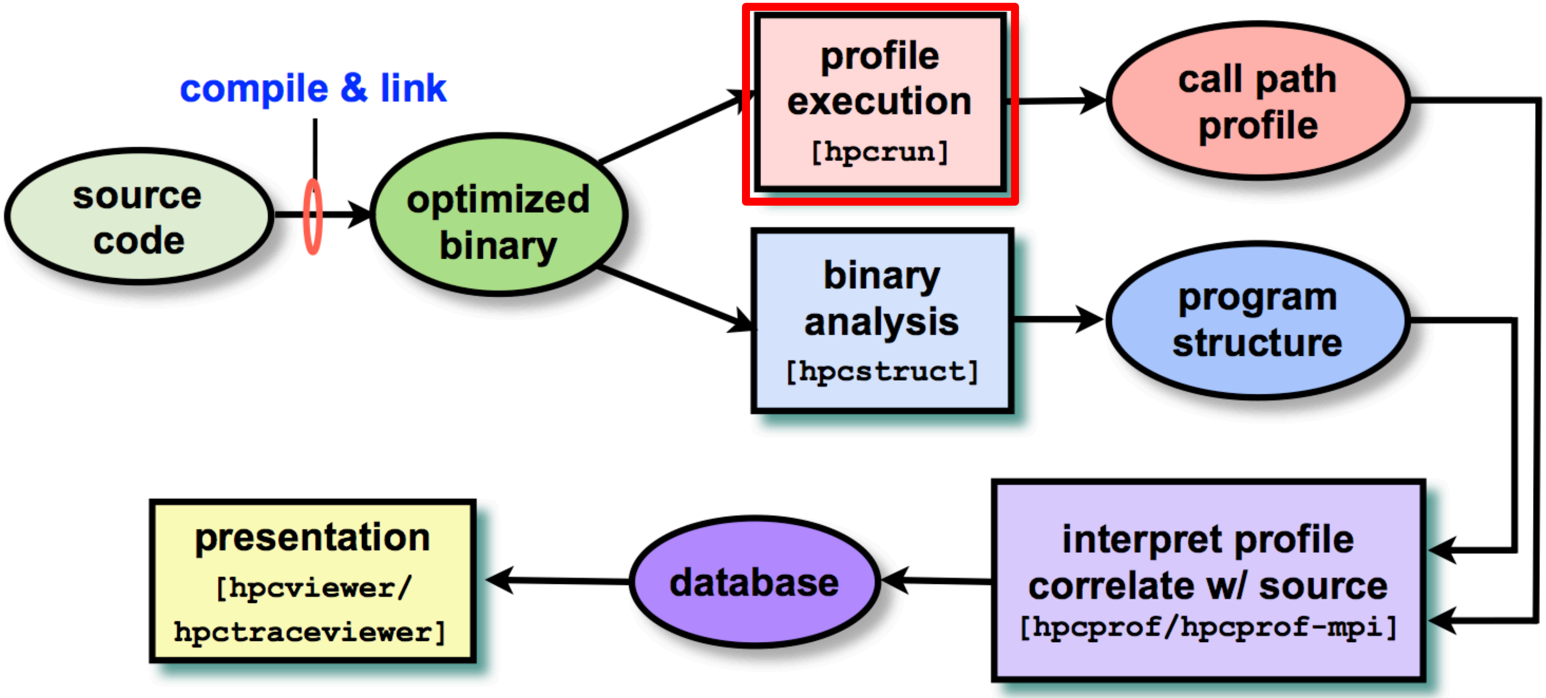
source: http://hpctoolkit.org

HPCToolkit Overview



source: http://hpctoolkit.org

HPCToolkit-hpcrun

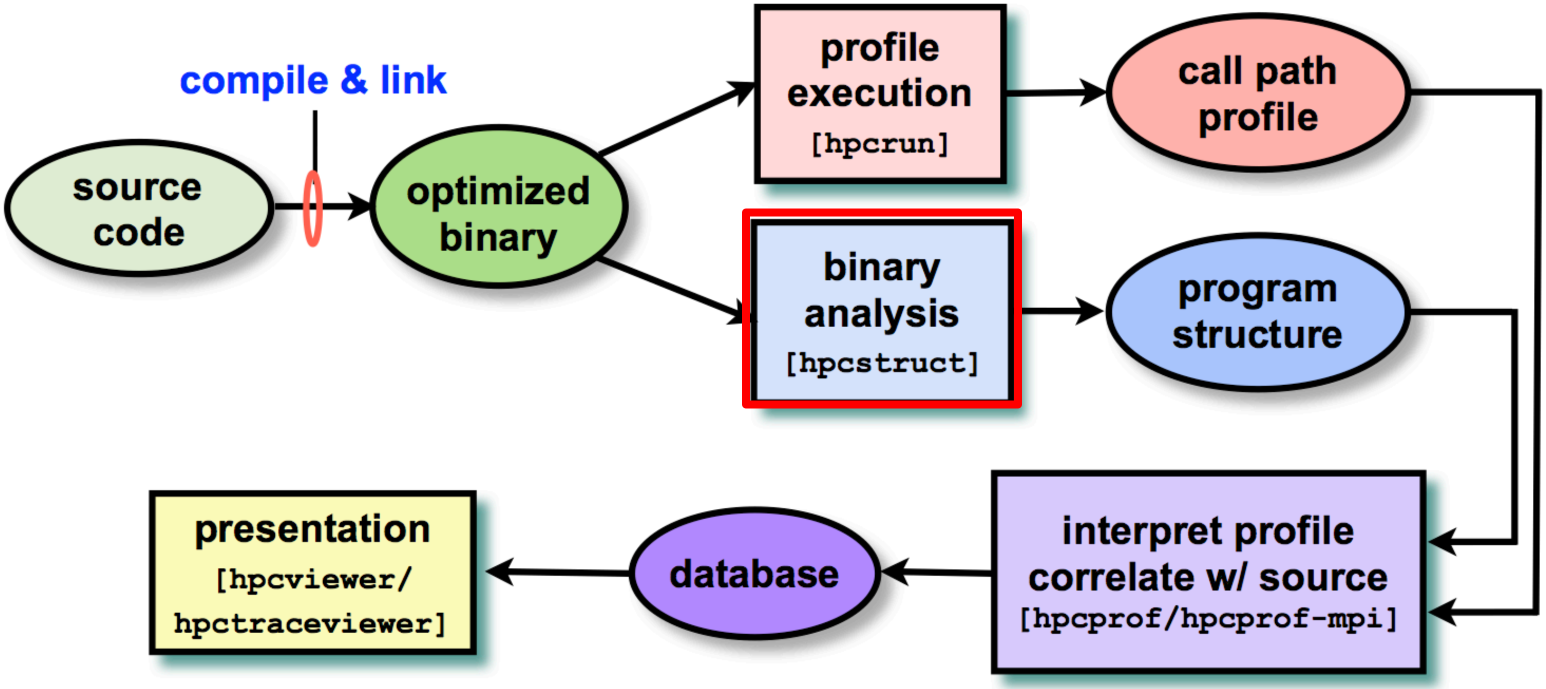


<u>hpcrun</u>

collecting calling-context-sensitive performance measurements

source: http://hpctoolkit.org

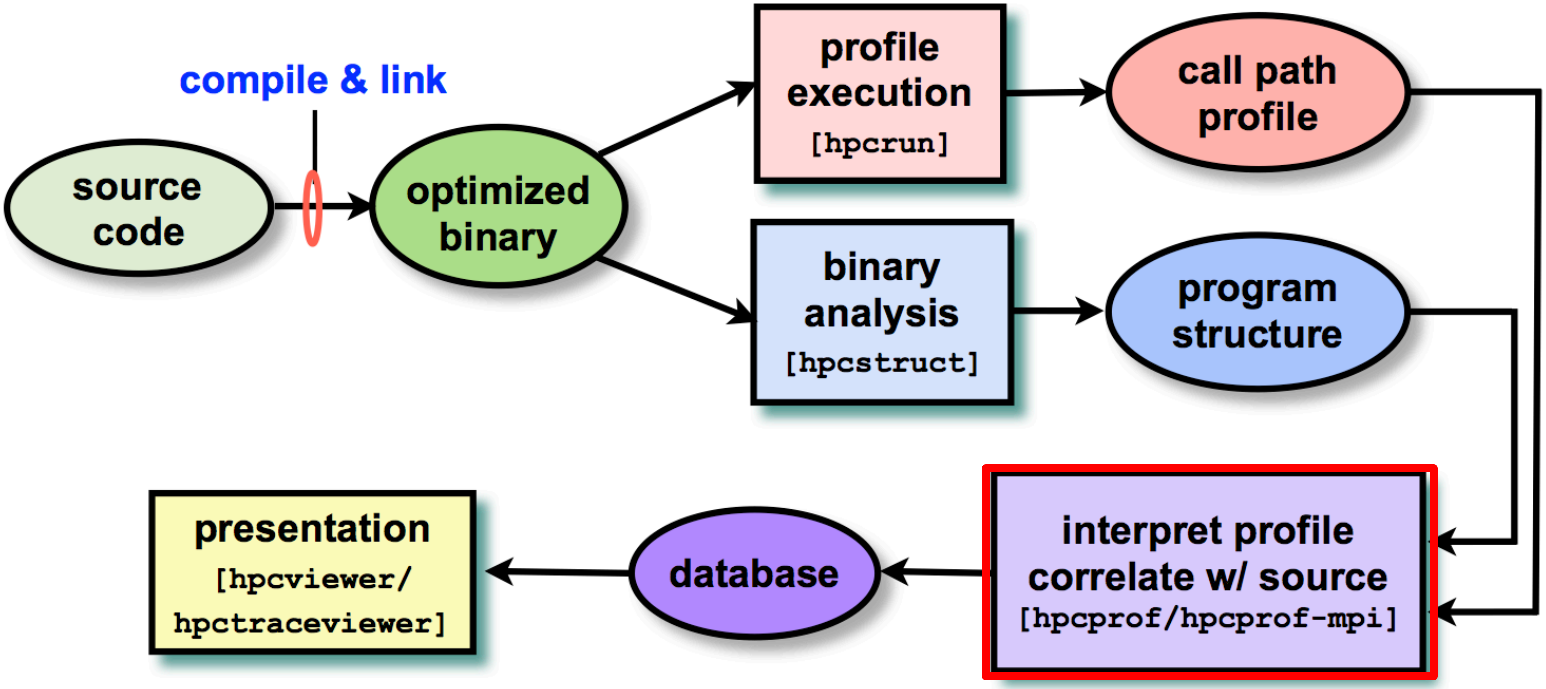
HPCToolkit-hpcstruct



<u>hpcstruct</u>

Analyzing application binaries and information between binaries and source codes

HPCToolkit - hpcprof

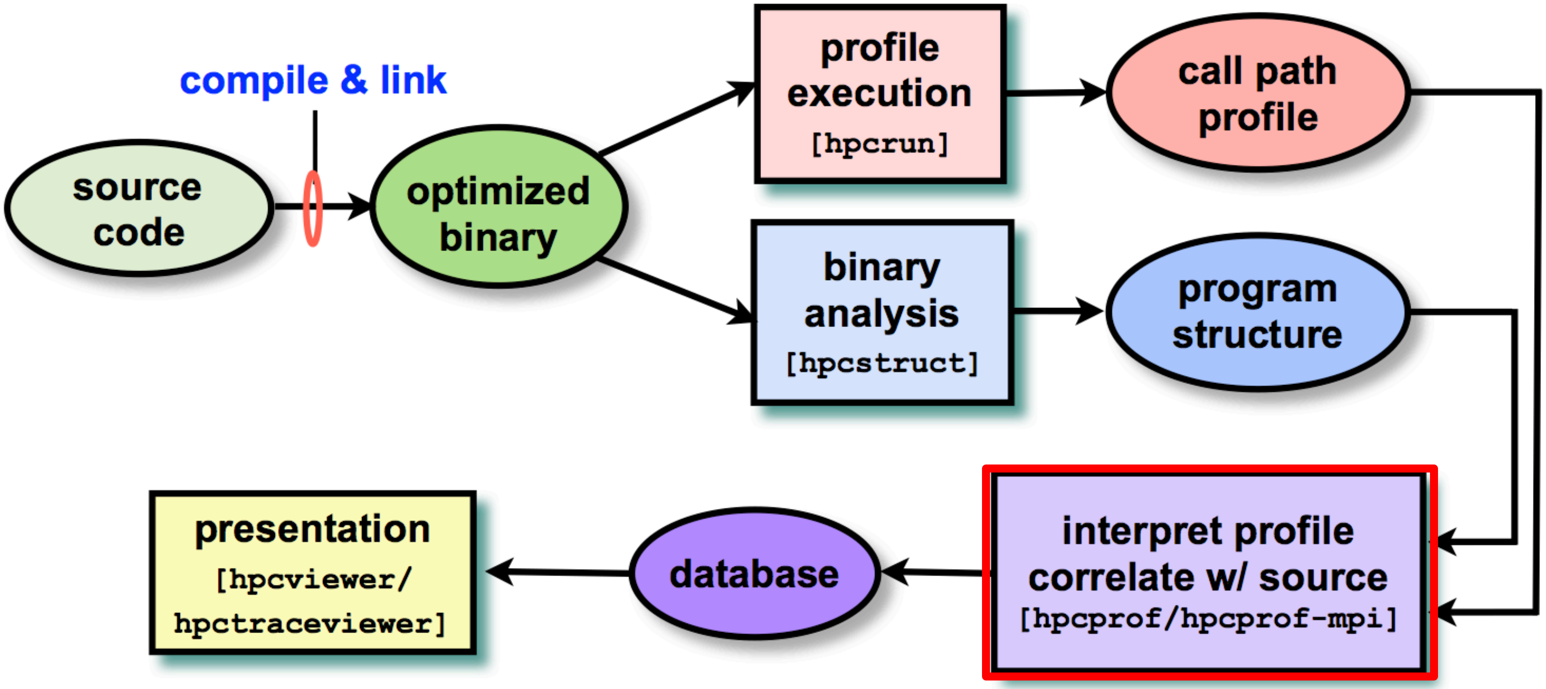


<u>hpcprof</u>

correlating the result with source code

source: http://hpctoolkit.org

HPCToolkit - hpcviewer



<u>hpcviewer</u>

graphical user interface that presents a hierarchical, time-centric view of a program execution

source: http://hpctoolkit.org

Example

- HPCToolkit example with Matmul
  - `> gcc -O2 -g -o gemm gemm.c`
  - `> hpcstruct ./gemm`
  - `> hpcrun-flat -e PAPI_TOT_CYC:500 ./gemm`
  - `> hpcproftt --src=p -S gemm.hpcstruct -I . gemm*0x0`

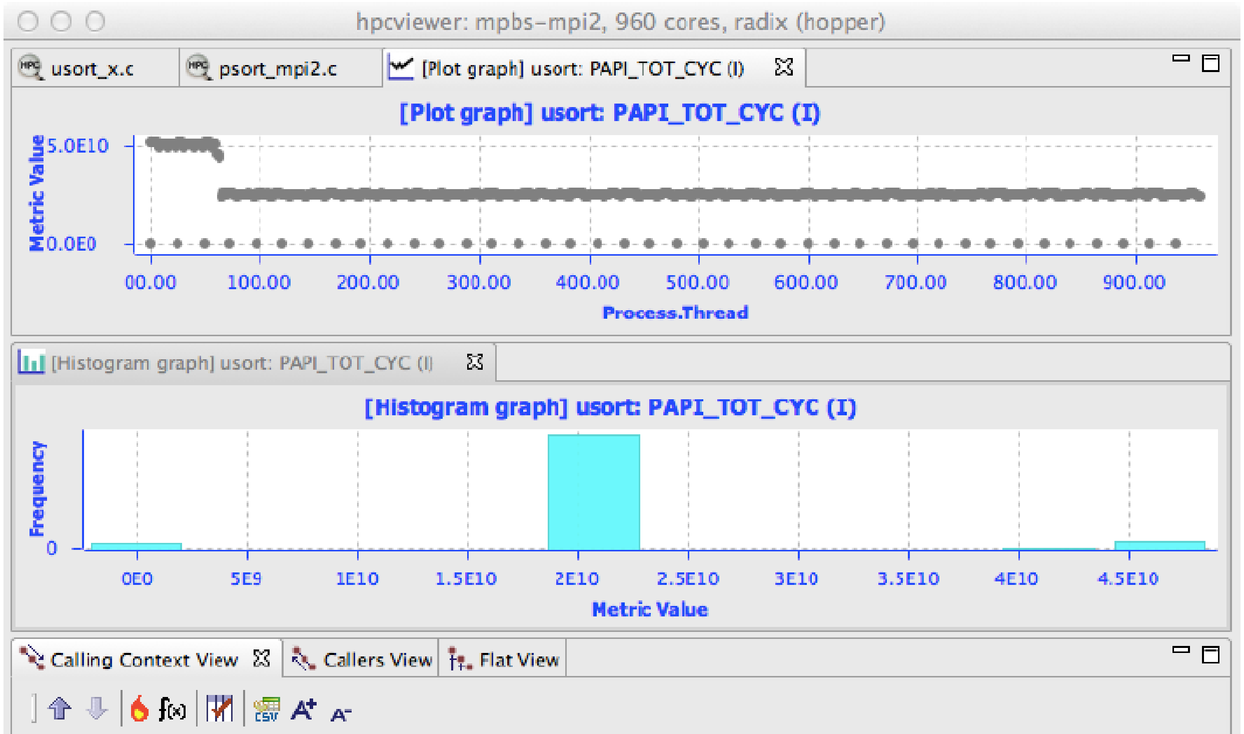
Output using hpcprof

- Text output using hpcprof

```
=========================================================
Metric definitions. column: name (nice-name) [units] {details}:
   1: PAPI_TOT_CYC [events] {Total cycles:500 ev/smpl}

================================================================
=Procedure summary:
----------------------------------------------------------------
-1.24e+09  […/gemm]<…/flush_cache.c>main
 5558500  […/gemm]<…/flush_cache.c>flush_cache
 3352000  […/gemm]<…/flush_cache.c>init_array
   72500  [/lib/ld-2.7.so]<~unknown-file~>_dl_rtld_di_serinfo
    5500  [/lib/ld-2.7.so]<~unknown-file~>realloc
       …    …
```

Visualization of Result



Scope	PAPI_TOT_CYC:Sum (I)		PAPI_TOT_CYC:Mean (I)	PAPI_TOT_CYC:StdDev (I)
Experiment Aggregate Metrics	5.18e+14	100 %	5.18e+11	1.54e+11
main	5.18e+14	100 %	5.18e+11	1.54e+11
604: rbyfls64	4.52e+14	87.3%	4.52e+11	1.45e+11
loop at ioops.c: 2314	4.52e+14	87.3%	4.52e+11	1.45e+11
2495: MPI_Barrier	3.11e+14	60.0%	3.11e+11	1.27e+11
MPIR_Barrier_impl	3.11e+14	60.0%	3.11e+11	1.27e+11
2365: psortui64_mpi2	1.21e+14	23.3%	1.21e+11	2.46e+10
loop at psort_mpi2.c: 801	5.04e+13	9.7%	5.04e+10	1.03e+10
862: usort	2.56e+13	5.0%	2.56e+10	8.08e+09

source: http://hpctoolkit.org

PerfExpert

- http://www.tacc.utexas.edu/perfexpert
- Tool for performance optimization for parallel architectures
- Automates most of intra-node performance optimization

source: http://www.tacc.utexas.edu/perfexpert

Goal of PerfExpert

- Automate detection and characterization of performance bottlenecks
- Suggest optimizations for each bottleneck
  - Including code examples and compiler switches
- Simplicity is paramount
  - Trivial user interface
  - Easily understandable output

source: http://www.tacc.utexas.edu/perfexpert

PerfExpert - Overview

- Gather performance counter measurements
  - Multiple runs with HPCToolkit
  - Sampling-based results for procedures and loops
- Combine results
  - Check variability, runtime, consistency, and integrity
- Compute and output assessment
  - Only for most important code sections
  - Correlate results from different thread counts

source: http://www.tacc.utexas.edu/perfexpert

Output of PerfExpert

```
total runtime in dgelastic.xml is 75.70 seconds

Suggestions on how to alleviate performance bottlenecks are available at:
              tacc.utexas.edu/perfexpert/

dgae_RHS (59.8% of the total runtime)
----------------------------------------------------------------------------
performance assessment   great.....good.....okay.....bad......problematic
- overall                >>>>>>>>>>>>>>>>>>>>>>>>>>>>>
upper bound by category
- data accesses          >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>+
- instruction accesses   >>>>>>>>>
- floating-point instr   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
- branch instructions    >>
- data TLB               >
- instruction TLB        >
```

procedure identifier

URL to suggested optimizations

overall loop performance is bad

most of runtime is due to data accesses

source: http://www.tacc.utexas.edu/perfexpert

Parallelism V
HPC Libraries

John Cavazos

Dept of Computer & Information Sciences

University of Delaware

Lecture Overview

- High Performance Parallel Libraries
  - BLAS
  - LAPACK
  - ATLAS
  - Intel MKL
  - ACML (AMD core math library)

BLAS

- Basic Linear Algebra System
  - Level 1 – vector-vector operations
  - Level 2 – matrix-vector operations
  - Level 3 – matrix-matrix operations
- Fundamental level of linear algebra libraries
- Various highly optimized implementations by vendors
- Many other libraries built on top of BLAS

BLAS Level 3 Example

- Using gsl_blas_dgemm function ($C \leftarrow \alpha AB + \beta C$)

```
int gsl_blas_dgemm(
     CBLAS_TRANSPOSE_t TransA,
     CBLAS_TRANSPOSE_t TransB,
     double alpha,
     const gsl_matrix * A,
     const gsl_matrix * B,
     double beta,
     gsl_matrix * C)
```

TransA: `CblasNoTrans` or `CblasTrans` for A
TransB: `CblasNoTrans` or `CblasTrans` for B

BLAS Level 3 Example

```
#include <stdio.h>
#include <gsl/gsl_blas.h>

int
main (void)
{
  double a[] = { … }; // a = 2x3 matrix
  double b[] = { … }; // b = 3x2 matrix
  double c[] = { … }; // c = 2x2 matrix

  gsl_matrix_view A = gsl_matrix_view_array(a, 2, 3);
  gsl_matrix_view B = gsl_matrix_view_array(b, 3, 2);
  gsl_matrix_view C = gsl_matrix_view_array(c, 2, 2);

  /* Compute C = A B */

  gsl_blas_dgemm (CblasNoTrans, CblasNoTrans,
                  1.0, &A.matrix, &B.matrix,
                  0.0, &C.matrix);
}
```

LAPACK

- Written on top of Basic Linear Algebra Subprograms (BLAS)
- Incorporates/retools EISPACK (eigenvalues) and LINPACK (least squares)
- Optimized for most modern shared memory architectures
- Website: http://www.netlib.org/lapack/

Eigenvector **Eigenvalue**

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 2 & 1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 6 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

LAPACK Problems Solved

- Systems of linear equations
- Linear least squares problems
- Eigenvalue problems
- Singular value problems
- Associated computations
  - Matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur)
  - Reordering of the Schur factorizations
  - Estimating condition numbers
  - ...

ScaLAPACK

- Parallel version of LAPACK
- Designed for distributed-memory message-passing MIMD computers and networks of workstations supporting PVM and MPI
- Designed for workstations, vector supercomputers, and shared-memory-parallel computers

source: http://acts.nersc.gov/scalapack/index.html

ATLAS

- Automatically Tuned Linear Algebra Software
- http://math-atlas.sourceforge.net
- Provides high performance dense linear algebra routines:
  - BLAS, some LAPACK
- Automatically adapts itself to differing architectures using empirical techniques

ATLAS

- Performs a series of timed tests upon installation.
- These tests are used to tune the libraries for the individual system.
- Substantially faster on many systems.

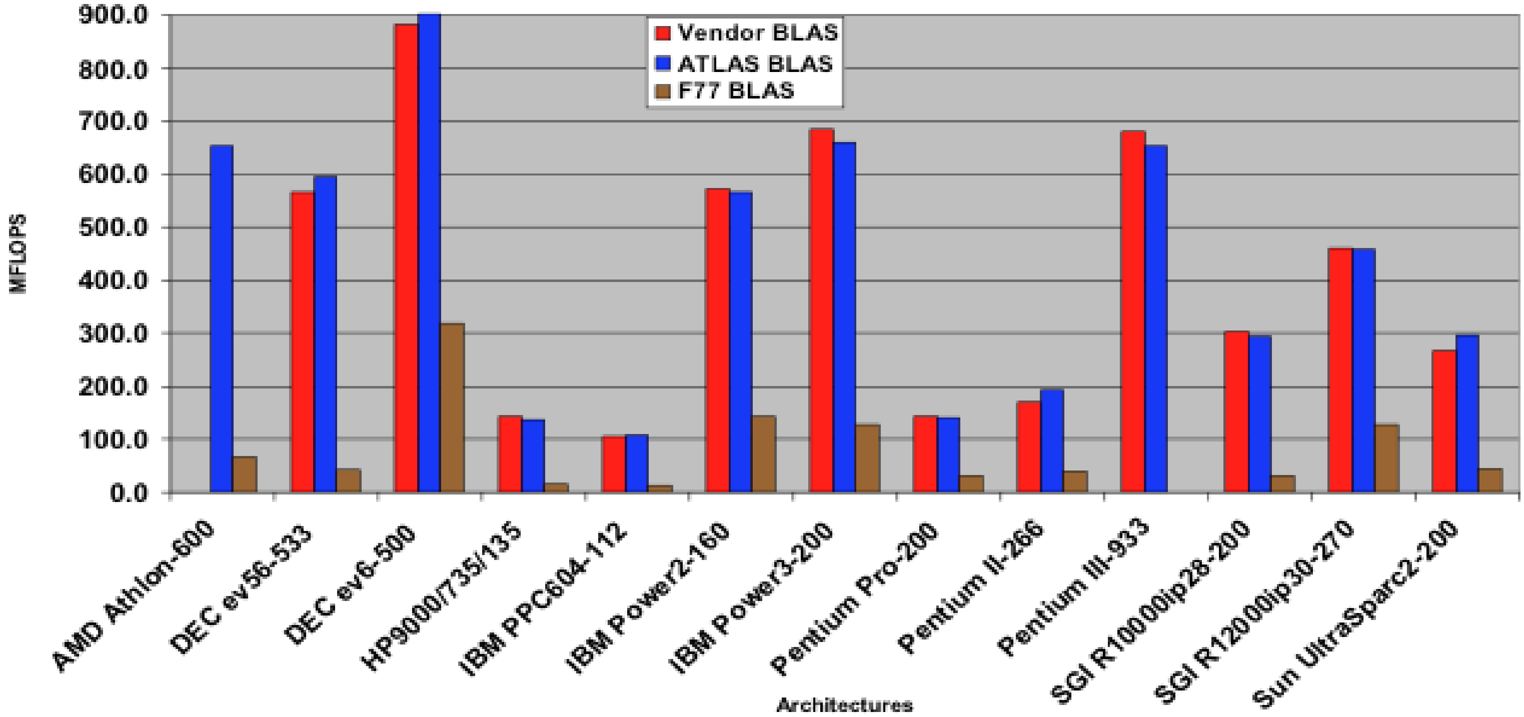
Why ATLAS?

- Well-tuned linear algebra routine runs orders of magnitude faster than generic implementation
- Hand-tuning is architecture specific
- No such thing as enough compute speed for many scientific codes

Present ATLAS tuning

- Written in ANSI C, output ANSI C
- Unrolling all loops
  - outer loops are jammed
- Different prefetch strategies
- Loop peeling
- Software pipelining
- Other backend optimizations
  - Register blocking, Inst scheduling, etc.

ATLAS Performance



ATLAS is faster than generic BLAS and as good as machine-specific libraries provided by vendor.

Intel MKL

- Intel Math Kernel Library
  - http://software.intel.com/en-us/intel-mkl/
  - Highly optimized and extensively threaded math library especially suitable for computational intensive applications.
  - Addresses numeric intensive codes in a broad range of disciplines in engineering, science and finance

MKL Contents

- BLAS (Basic Linear Algebra Subroutines)
- Extended BLAS- Level 1 BLAS for sparse vectors
- LAPACK (Linear Algebra Package)
  - Hundreds of routines for solvers and eigensolvers
- Fortran

Example

- Matrix-Vector Multiplication ($Y \leftarrow \alpha AB + \beta C$)

```
for( i = 0; i < n; i++ )
     cblas_dgemv( CBLAS_RowMajor, CBLAS_NoTrans, \
     m, n, alpha, a, lda, &b[0][i], ldb, beta, \
  &c[0][i], ldc );
```

- Matrix-Matrix Multiplication ($C \leftarrow \alpha AB + \beta C$)

```
Cblas_dgemm( CblasColMajor, CblasNoTrans,
CblasNoTrans, m, n, kk, alpha, b, ldb, a, lda,
beta, c, ldc );
```

ACML

- AMD Core Math Library
- Very similar to Intel MKL!
- Provides developers of scientific and engineering software with
  - A set of linear algebra
  - Fast Fourier transforms
  - Vector math functions
  - LAPACK, BLAS, and the extended BLAS
- Installed on Mills cluster