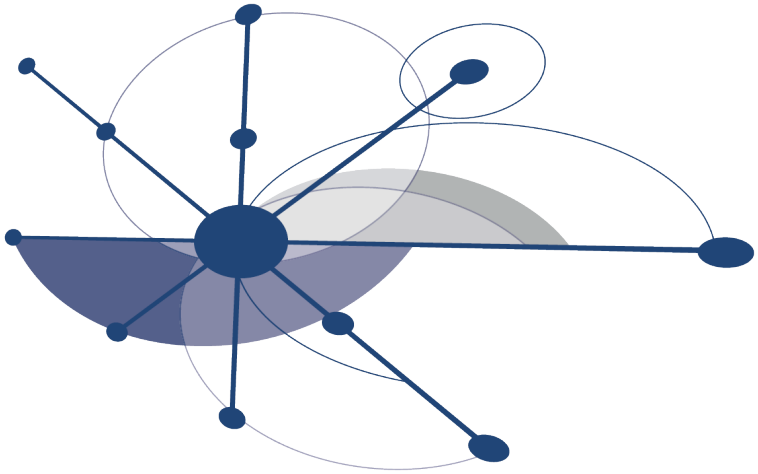


Analyzing the Performance of IWAVE on a Cluster using HPCToolkit

John Mellor-Crummey
Department of Computer Science
Rice University
johnmc@rice.edu

SciDAC
Scientific Discovery through Advanced Computing

RICE

Challenges for HPC Practitioners

- Execution environments and applications are rapidly evolving
  - architecture
    - rapidly changing multicore microprocessor designs
    - increasing scale of parallel systems
    - growing use of accelerators, e.g. GPGPU
  - applications
    - MPI everywhere to threaded implementations
    - adding additional scientific capabilities to existing applications
    - maintaining multiple variants or configurations for particular problems
- Steep increase in application development effort to attain performance, evolvability, and portability
- Application developers need to
  - assess weaknesses in algorithms and their implementations
  - improve scalability and performance within and across nodes
  - adapt to changes in emerging architectures
  - overhaul algorithms & data structures as needed

Performance tools can play an important role as a guide

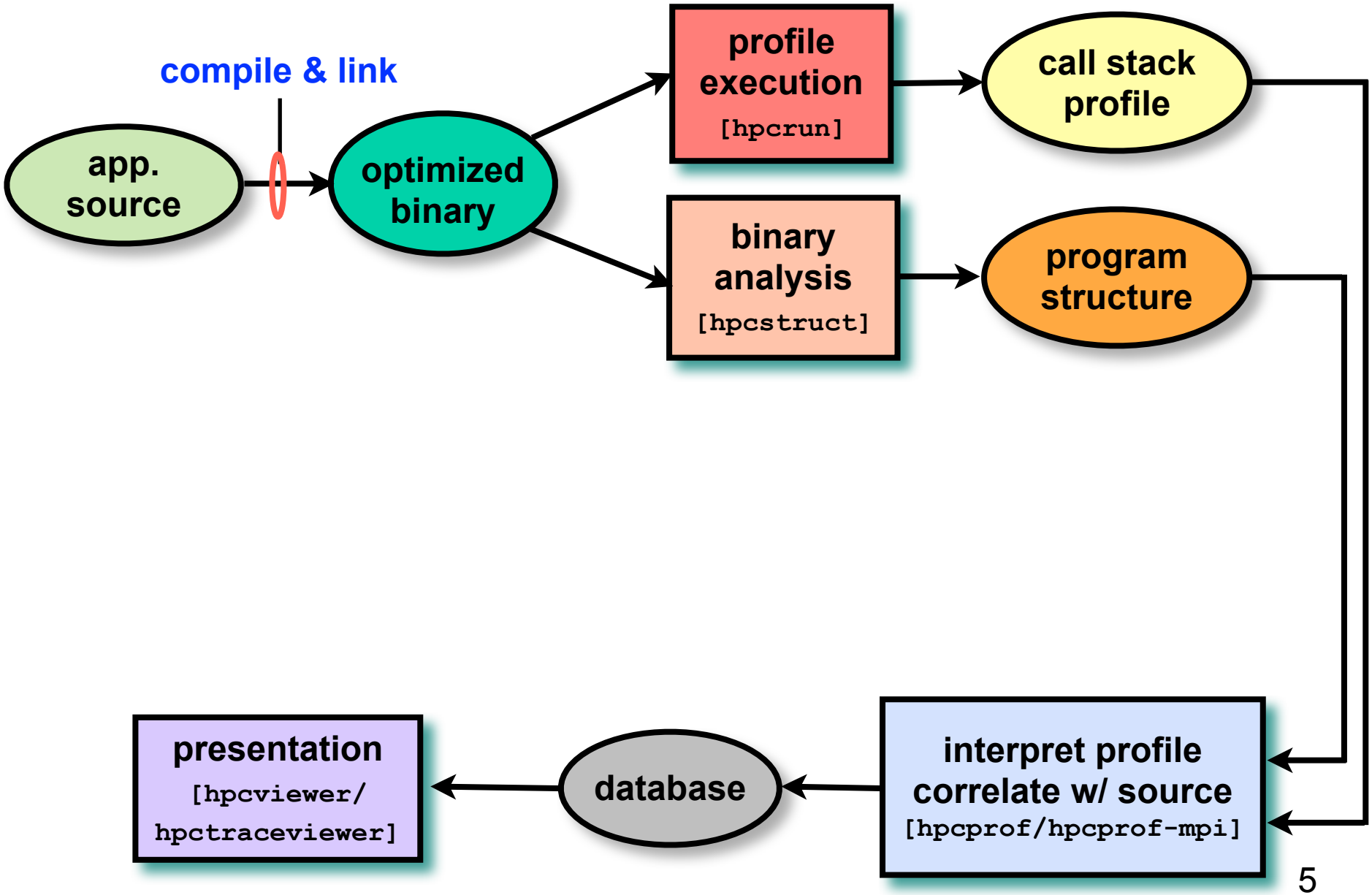
Performance Analysis Challenges

- **Complex architectures are hard to use efficiently**
  - multi-level parallelism: multi-core, ILP, SIMD instructions
  - multi-level memory hierarchy
  - result: gap between typical and peak performance is huge
- **Multifaceted performance concerns**
  - computation
  - communication
  - I/O
- **Complex applications present challenges**
  - for measurement and analysis
  - for understanding and tuning

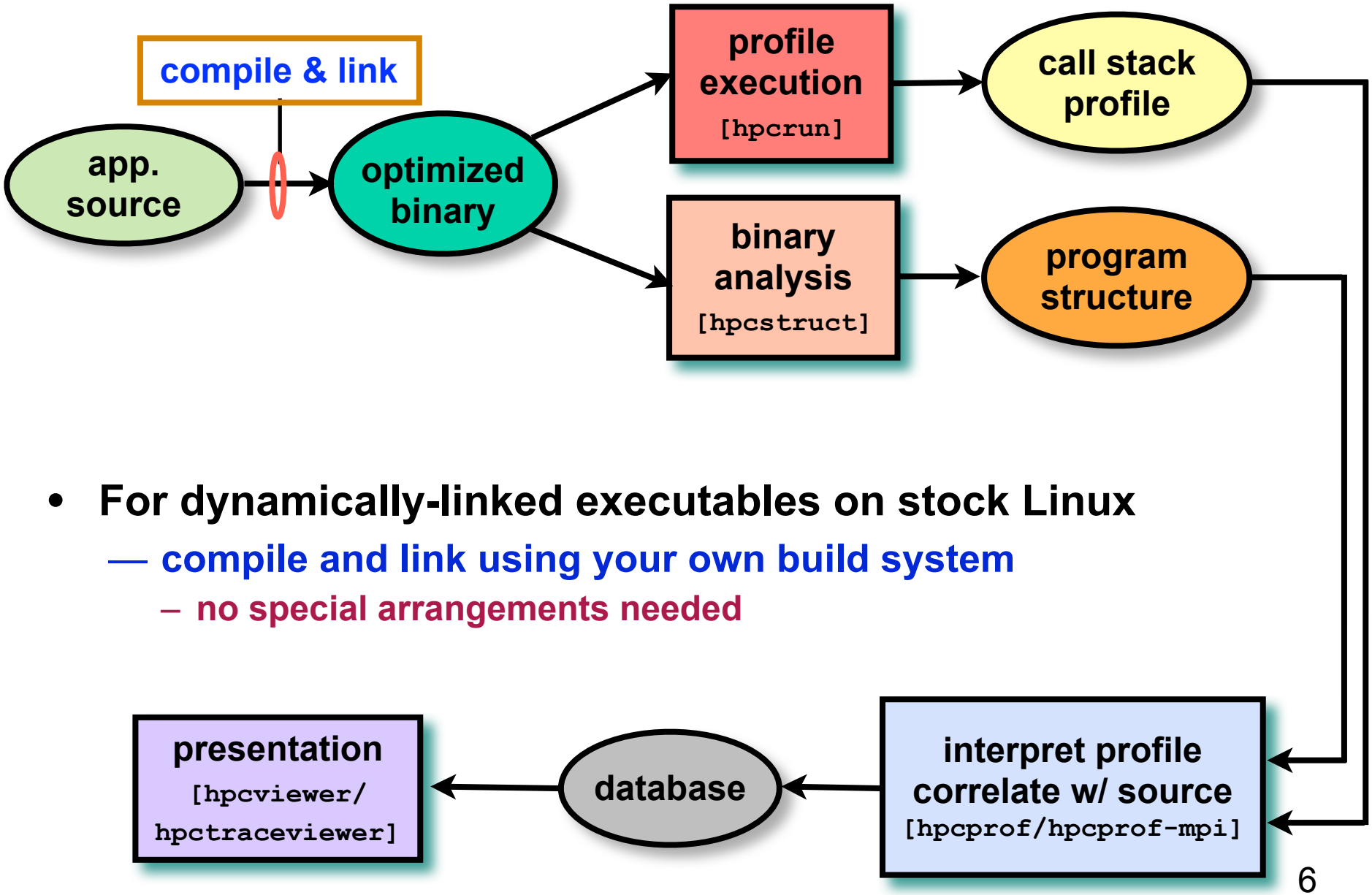
Performance Analysis Goals

- Programming model independent tools
- Accurate measurement of complex parallel codes
  - large, multi-lingual programs
  - fully optimized code: loop optimization, templates, inlining
  - binary-only libraries, sometimes partially stripped
  - complex execution environments
    - dynamic loading (Linux clusters) vs. static linking (Cray, Blue Gene)
    - SPMD parallel codes with threaded node programs
    - batch jobs
- Effective performance analysis
  - insightful analysis that pinpoints and explains problems
    - correlate measurements with code for actionable results
    - support analysis at the desired level
      - intuitive enough for application scientists and engineers
      - detailed enough for library developers and compiler writers
- Scalable to petascale systems and beyond

HPCToolkit Workflow

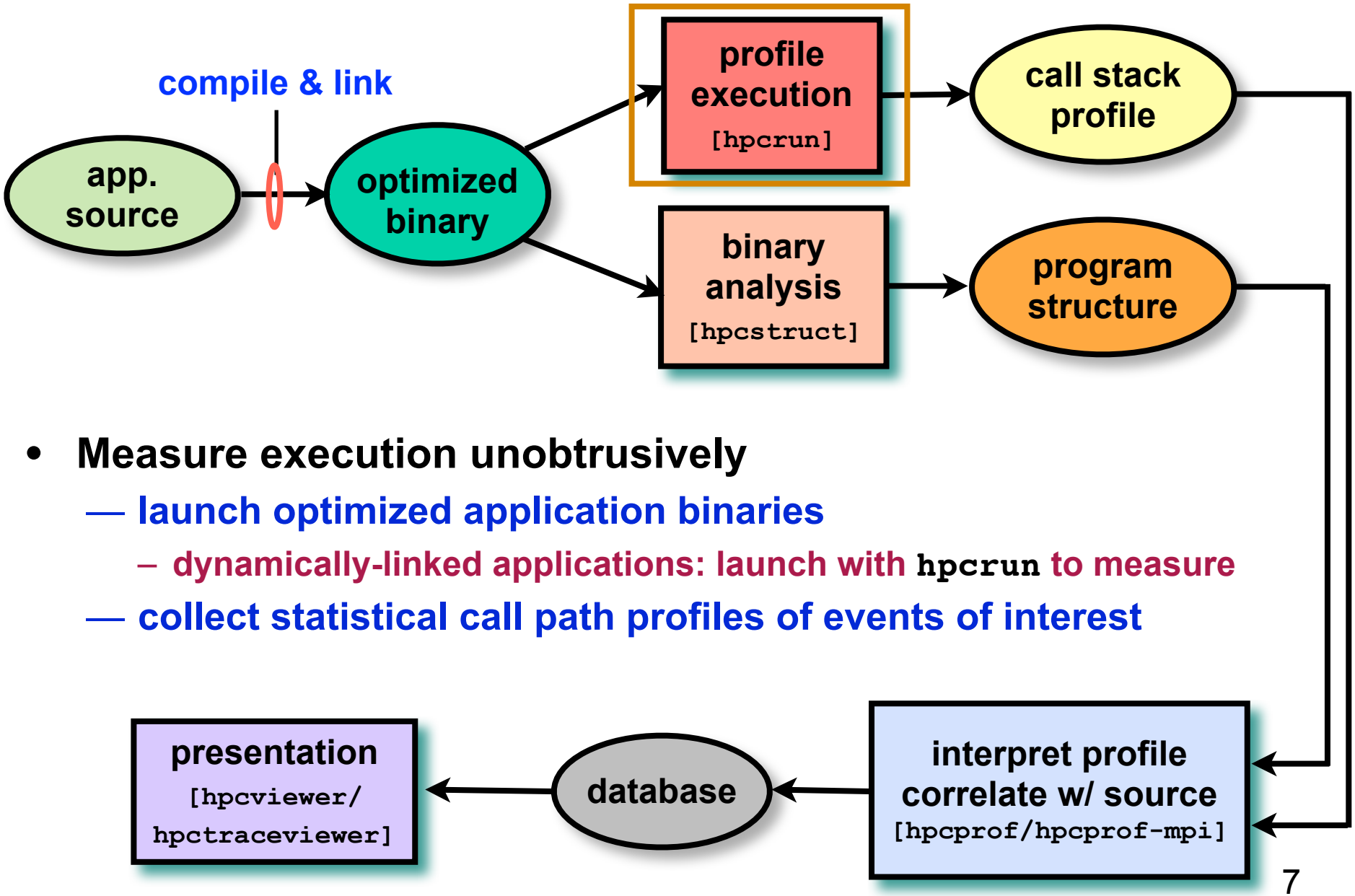


HPCToolkit Workflow



- For dynamically-linked executables on stock Linux
  - compile and link using your own build system
    - no special arrangements needed

HPCToolkit Workflow



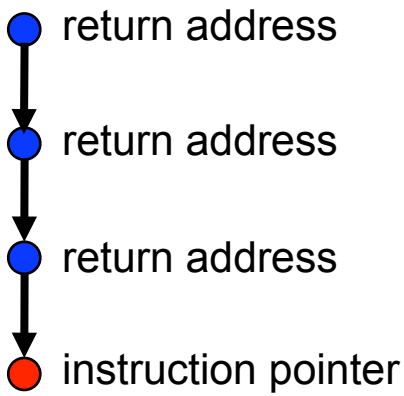
- **Measure execution unobtrusively**
  - launch optimized application binaries
    - dynamically-linked applications: launch with `hpcrun` to measure
  - collect statistical call path profiles of events of interest

Call Path Profiling

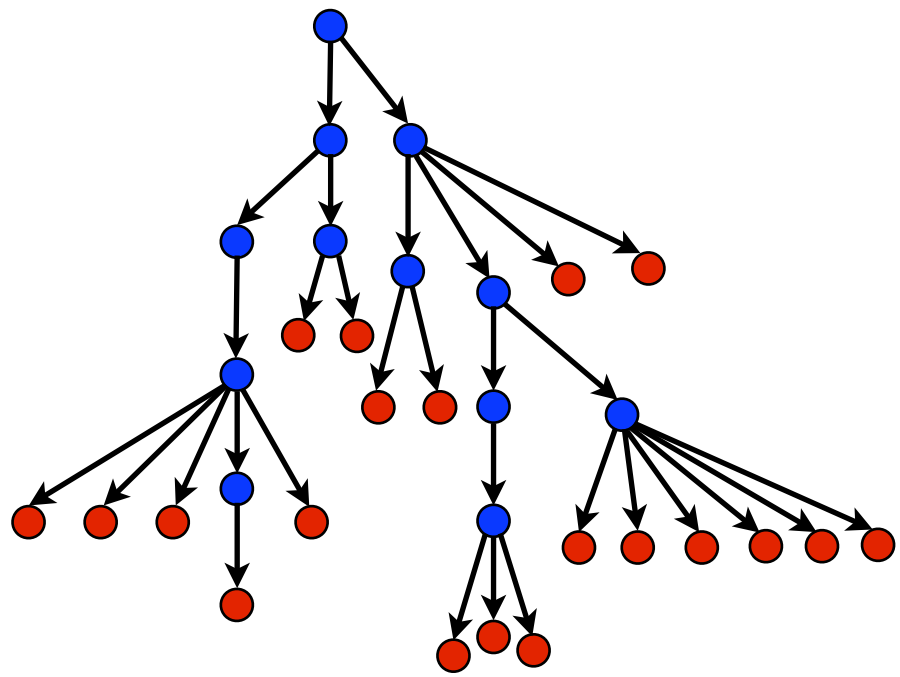
Measure and attribute costs in context

- **sample timer or hardware counter overflows**
- **gather calling context using stack unwinding**

Call path sample

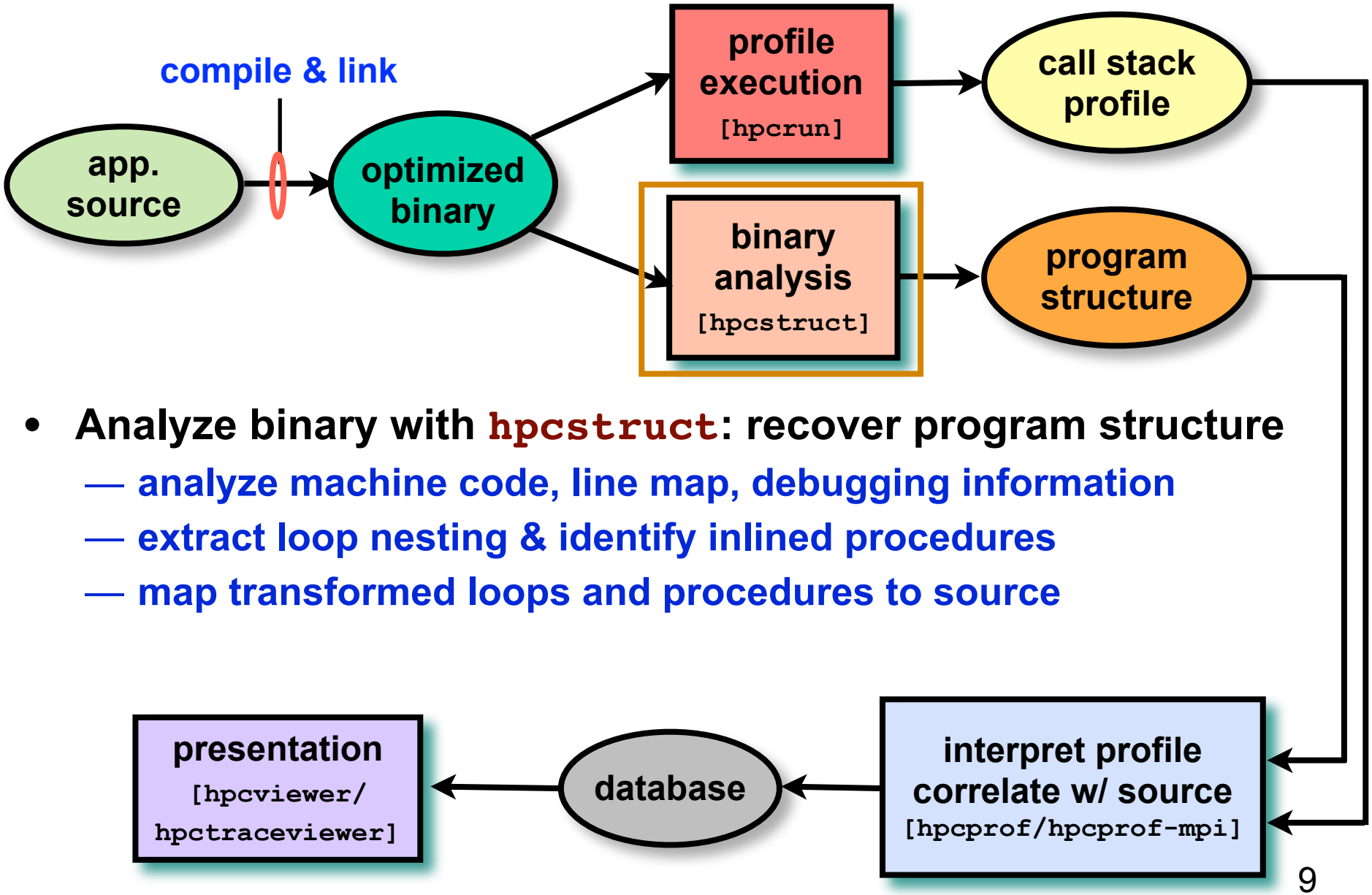


Calling context tree



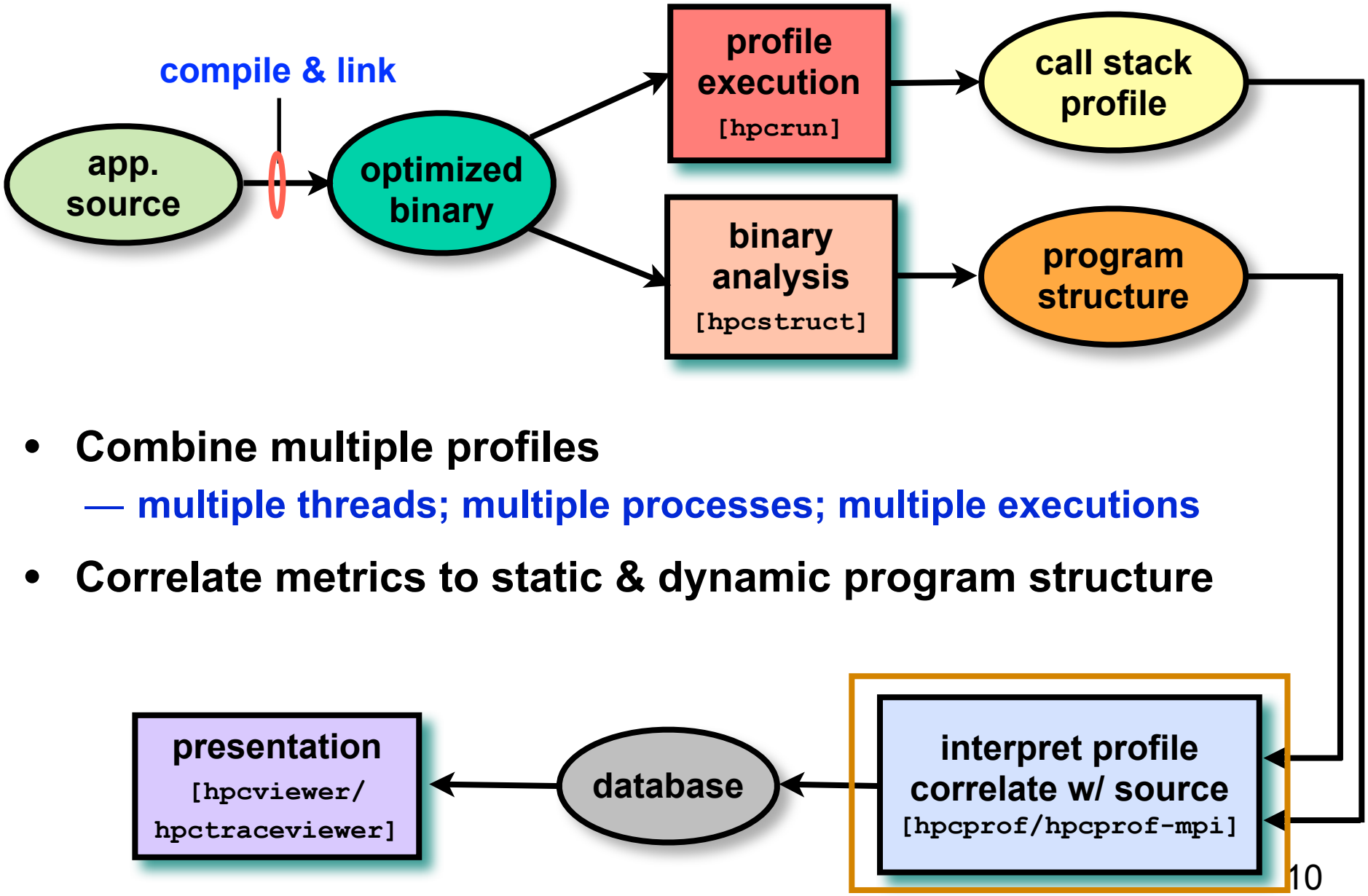
Overhead proportional to sampling frequency...
...not call frequency

HPCToolkit Workflow



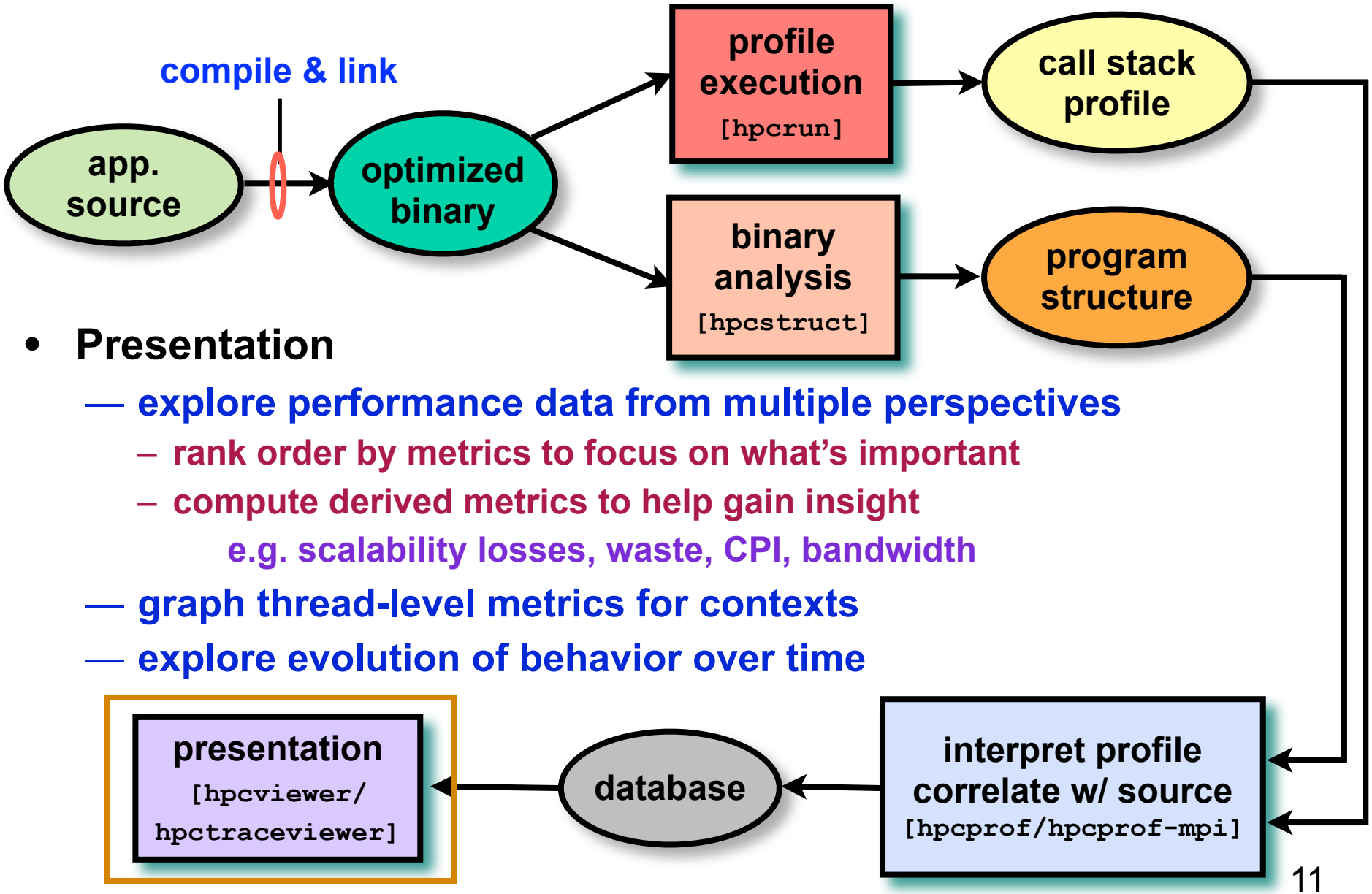
- Analyze binary with `hpcstruct`: recover program structure
  - analyze machine code, line map, debugging information
  - extract loop nesting & identify inlined procedures
  - map transformed loops and procedures to source

HPCToolkit Workflow



- Combine multiple profiles
  - — multiple threads; multiple processes; multiple executions
- Correlate metrics to static & dynamic program structure

HPCToolkit Workflow



- **Presentation**
  - — explore performance data from multiple perspectives
    - – rank order by metrics to focus on what's important
    - – compute derived metrics to help gain insight
      - e.g. scalability losses, waste, CPI, bandwidth
  - — graph thread-level metrics for contexts
  - — explore evolution of behavior over time

Novel Aspects of Our Approach

- **Unwind fully-optimized and even stripped code**
  - use on-the-fly binary analysis to support unwinding
- **Cope with dynamically-loaded shared libraries on Linux**
  - note as new code becomes available in address space
  - problematic for instrumentation-based tools
- **Integrate static & dynamic context information in presentation**
  - dynamic call chains including procedures, inlined functions, loops, and statements

Code-centric Analysis with `hpcviewer`

hpcviewer: MOAB: mbperf_iMesh 200 B (Barcelona 2360 SE)

mbperf_iMesh.cpp | TypeSequenceManager.hpp | stl_tree.h

```
22   * Define less-than comparison for EntitySequence pointers as a comparison
23   * of the entity handles in the pointed-to EntitySequence
24   */
25  class SequenceCompare {
26    public: bool o ntitySequence* a, const EntitySequence* b ) const
27      { return a->start_handle(); }
28  };
```

source pane

costs for
- inlined procedures
- loops
- function calls in full context

Calling Context View | Callers View | Flat View

view control

metric display

navigation pane

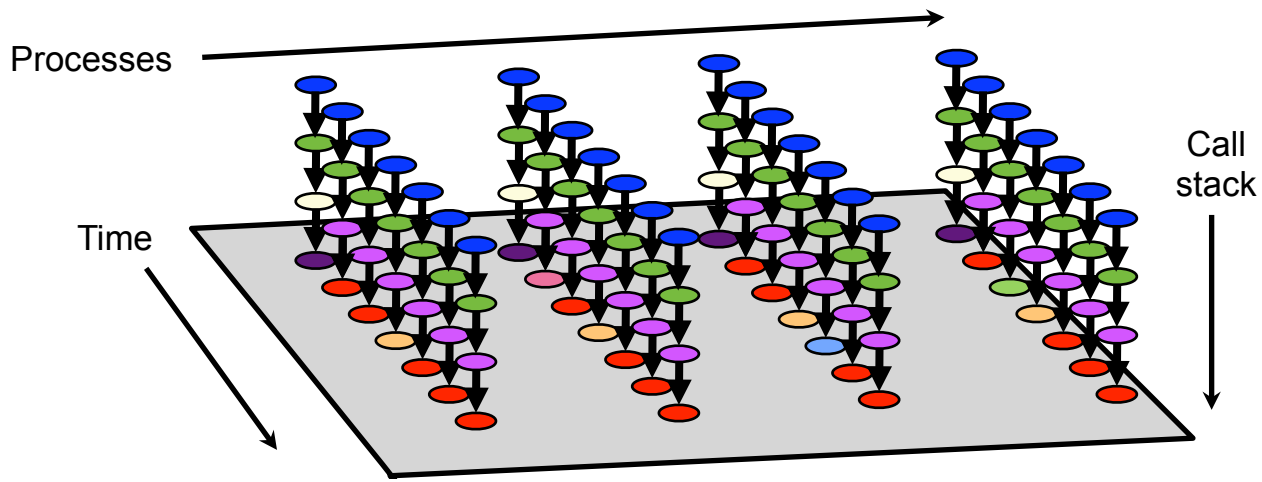
metric pane

Scope	PAPI_L1_DCM (I)		PAPI_TOT_CYC (I)	
main	8.63e+08	100 %	1.13e+11	100 %
testB(void*, int, double const*, int const*)	8.35e+08	96.7%	1.10e+11	97.6%
inlined from mbperf_iMesh.cpp: 261	6.81e+08	78.9%	0.98e+11	86.5%
loop at mbperf_iMesh.cpp: 280-313	3.43e+08	[illegible]	[illegible]	[illegible].9%
imesh_getvtxarrcoords_	3.20e+08	37.1%	2.18e+10	19.3%
MBCore::get_coords(unsigned long const*, int, double*) cc	3.20e+08	37.1%	2.16e+10	19.1%
loop at MBCore.cpp: 681-693	3.20e+08	37.1%	2.16e+10	19.1%
inlined from stl_tree.h: 472	2.04e+08	23.7%	9.38e+09	8.3%
loop at stl_tree.h: 1388	2.04e+08	23.6%	9.37e+09	8.3%
inlined from TypeSequenceManager.hpp: 27	1.78e+08	20.6%	8.56e+09	7.6%
TypeSequenceManager.hpp: 27	1.78e+08	20.6%	8.56e+09	7.6%

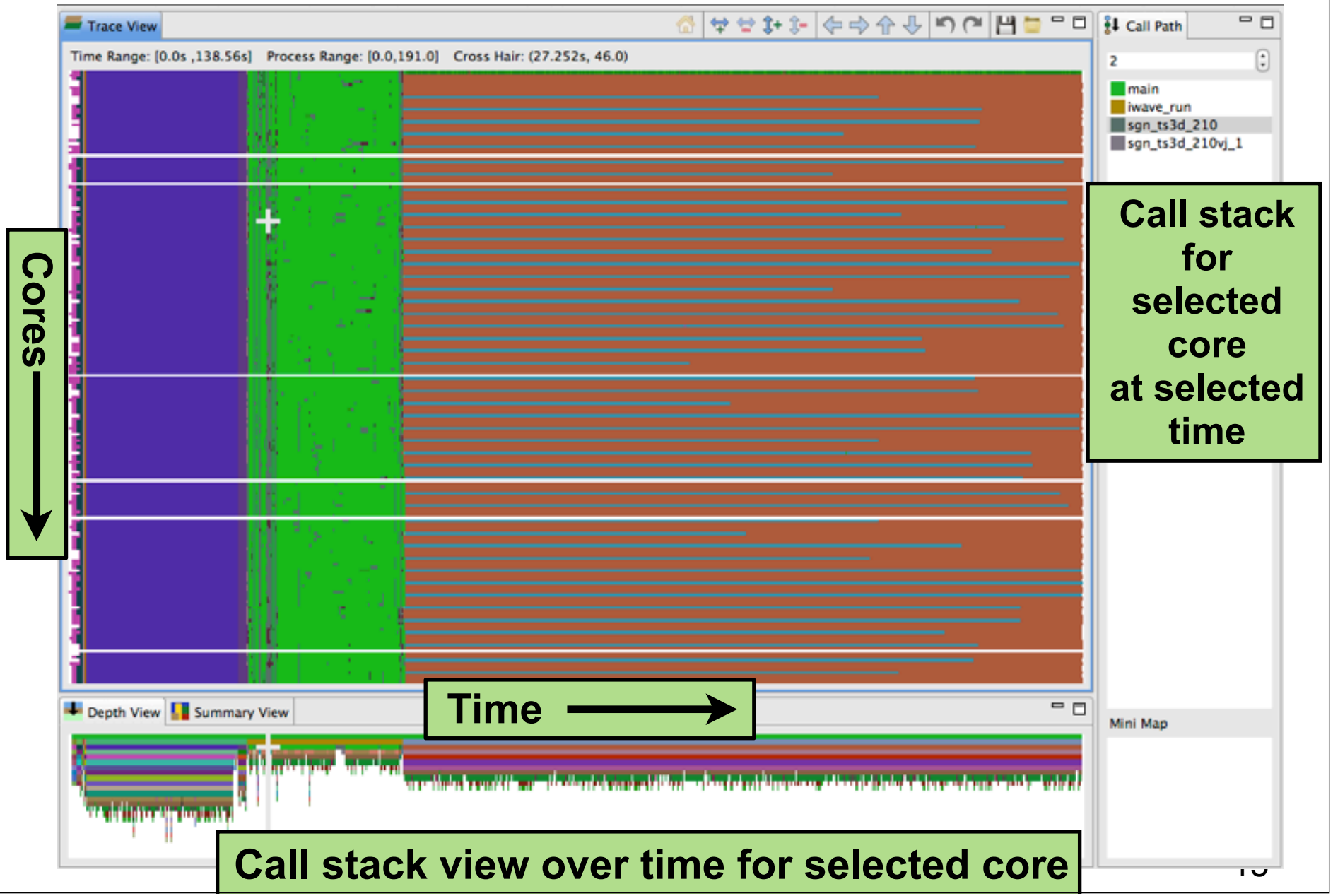
Understanding Executions over Time

- **Profiling compresses out the temporal dimension**
  - —temporal patterns, e.g. serialization, are invisible in profiles
- **What can we do? Trace call path samples**
  - —sketch:
    - N times per second, take a call path sample of each thread
    - organize the samples for each thread along a time line
    - view how the execution evolves left to right
    - what do we view?

      assign each procedure a color; view a depth slice of an execution



HPCToolkit's Time-centric View



IWAVE - Rice Inversion Project (Symes, PI)

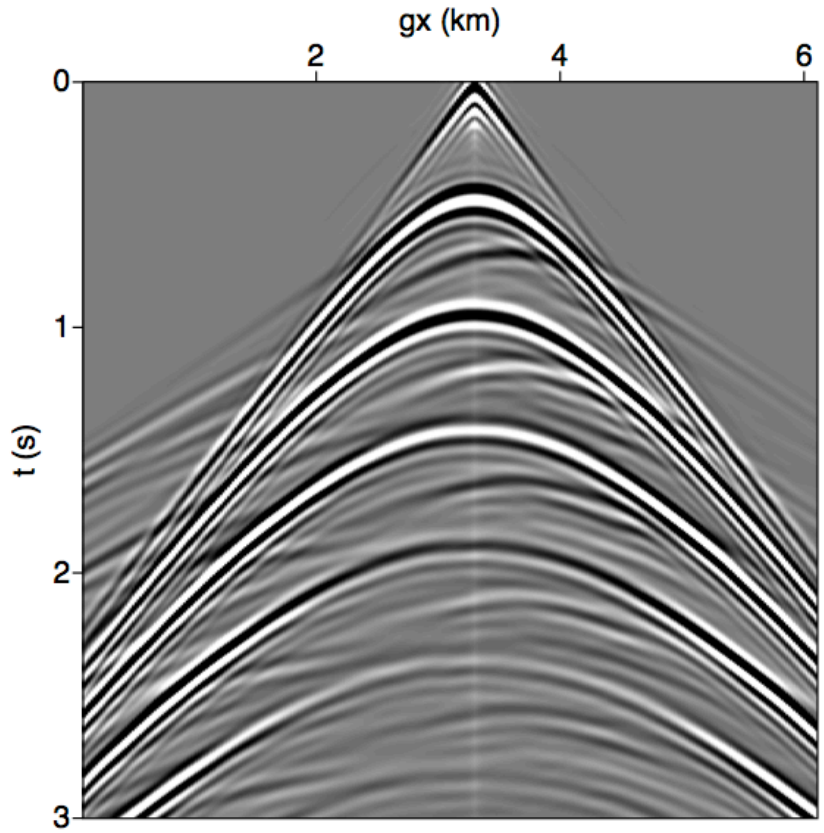
- **Framework for finite difference simulation**
  - **common services - memory, communication, I/O, job control**
  - **prescribed interfaces - problem description, numerical schemes**
- **Applications written to the framework**
  - **staggered grid acoustics with PML**
  - **staggered grid isotropic elasticity with PML**
- **Portable - ISO C99, MPI, OpenMP**
- **Modeling engine for migration and inversion**

IWAVE on a Cluster

- **Experimental Platforms**
  - DaVinci
    - node: two 2.83 GHz Intel Westmere (6-core) processors, 48GB RAM
    - interconnect: 4x QDR Inifinband (40Gb/s)
    - GPFS fast scratch
  - STIC
    - node: two 2.4 GHz Intel Nehalem (4-core), 12GB RAM
    - interconnect: 2 4x DDR Infiniband links per node (20Gb/s each)
    - 11TB Panassas scratch
- **IWAVE configuration studied**
  - asg package
    - staggered grid finite difference (pressure, velocity) acoustic modeling
  - 3D finite difference configuration
  - compiled with icc, version 12.0.0
    - -O3 -std=c99 -g
  - SEAM 20M GRID, FOR SHOT S020433

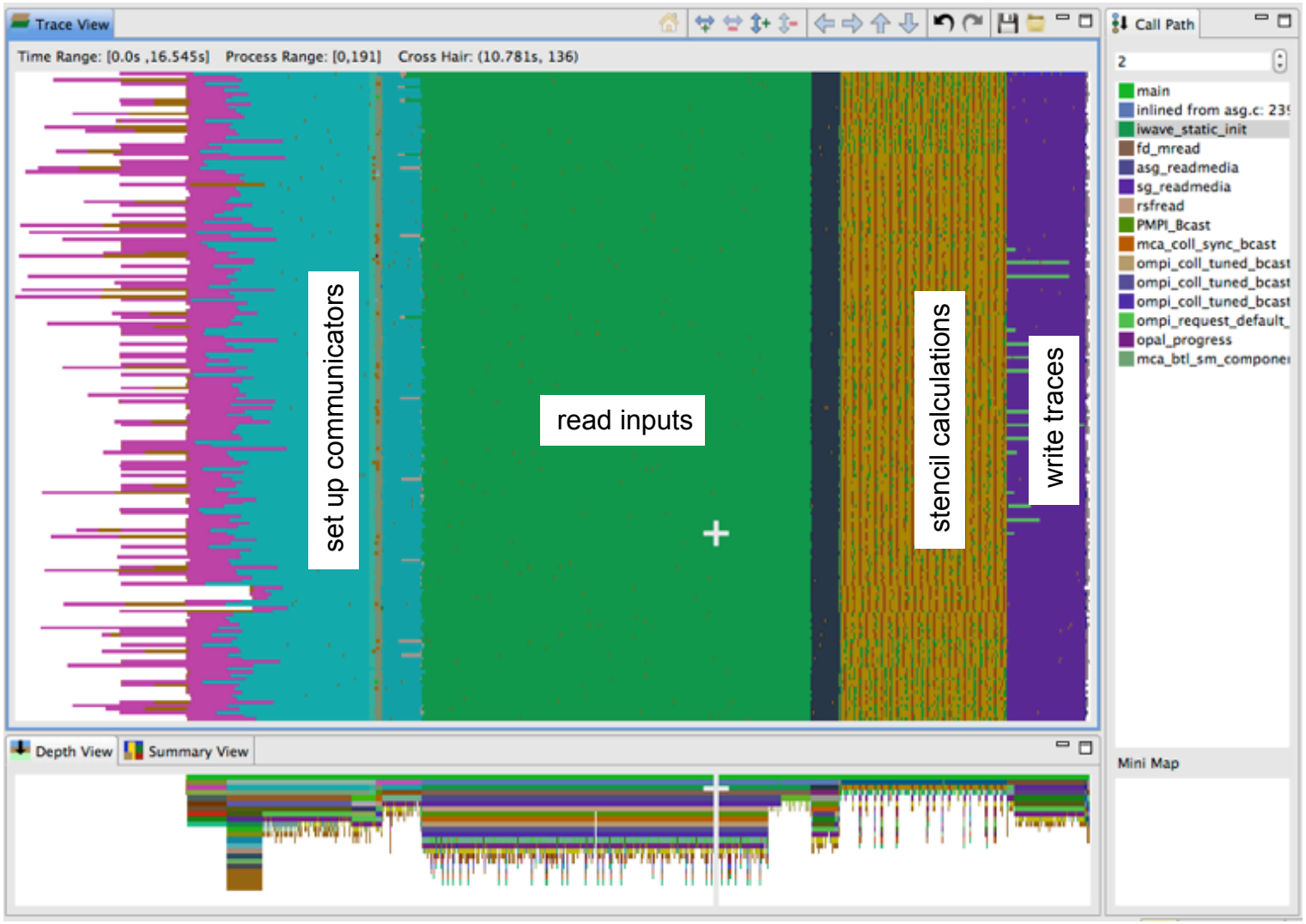
IWAVE Execution

- 8 x 6 x 4 MPI decomposition
- Model info
  - SEAMHALF20M
    - density info 808MB
    - velocity info 808MB
- IWAVE run
  - read velocity model
    - broadcast to all processors
  - read density model
    - broadcast to all processors
  - perform stencil calculations to compute pressures and velocities
  - write traces to disk



Time-centric view of IWAVE

- **MPI decomposition 8 x 6 x 4**
- **32 nodes, 6 cores each (192 processor cores), OpenMPI**



IWAVE Stencil - Overall Performance

hpcviewer: asg.x

sgn210_3d.c

```
256 #pragma ivdep
257         for ( _pxend = _px + nx; _px < _pxend; )
258           {
259         vx8 = _vx9[-1]; vx7 = _vx9[-2]; vx6 = _vx9[-3];
260         vx5 = _vx9[-4]; vx4 = _vx9[-5]; vx3 = _vx9[-6];
261         vx2 = _vx9[-7]; vx1 = _vx9[-8]; vx0 = _vx9[-9];
262
263         vx9 = *_vx9++;
264
265         delta = ((vx9 - vx0) * C5x + (vx8 - vx1) * C4x + (vx7 - vx2) * C3x + (vx6 - vx3) * C2x + (vx5 - vx4) * C1x +
266              ((*_vy9++) - (*_vy0++)) * C5y + ((*_vy8++) - (*_vy1++)) * C4y + ((*_vy7++) - (*_vy2++)) * C3y +
267              ((*_vy6++) - (*_vy3++)) * C2y + ((*_vy5++) - (*_vy4++)) * C1y +
268              ((*_vz9++) - (*_vz0++)) * C5z + ((*_vz8++) - (*_vz1++)) * C4z + ((*_vz7++) - (*_vz2++)) * C3z +
269              ((*_vz6++) - (*_vz3++)) * C2z + ((*_vz5++) - (*_vz4++)) * C1z) * (*_mpx++);
270
271         (*_px) = (*_px) * (*ratio_x++) + delta * (*recip_x++);
272         _px++;
273
274         (*_py) = (*_py) * ratio_y + delta * recip_y;
275         _py++;
276
277         (*_pz) = (*_pz) * ratio_z + delta * recip_z;
278         _pz++;
279           }
```

Calling Context View | Callers View | Flat View

Scope	PAPI_TOT_CYC:Sum (I)	PAPI_L2_TCM:Sum (I)	PAPI_TLB_DM:Sum (I)	PAPI_FP_INS:Sum (I)
Experiment Aggregate Metrics	1.36e+12 100 %	1.42e+10 100 %	4.10e+08 100 %	5.01e+11 100 %
main	1.36e+12 100.0	1.42e+10 100.0	4.10e+08 100 %	5.01e+11 100 %
loop at asg.c: 133	1.36e+12 100.0	1.42e+10 100.0	4.10e+08 100 %	5.01e+11 100 %
loop at asg.c: 239	1.36e+12 100.0	1.42e+10 100.0	4.10e+08 100 %	5.01e+11 100 %
loop at asg.c: 239	1.36e+12 100.0	1.42e+10 100.0	4.10e+08 100 %	5.01e+11 100 %
iwave_run	1.36e+12 100.0	1.42e+10 99.9%	4.10e+08 99.9%	5.01e+11 100.0
sgn_ts3d_210	8.83e+11 64.7%	1.20e+10 84.3%	2.15e+08 52.4%	5.00e+11 99.8%
sgn_ts3d_210p012	3.66e+11 26.8%	4.87e+09 34.2%	9.54e+07 23.3%	1.82e+11 36.2%
loop at sgn210_3d.c: 242	3.65e+11 26.8%	4.86e+09 34.1%	9.50e+07 23.2%	1.82e+11 36.2%
loop at sgn210_3d...	3.65e+11 26.8%	4.86e+09 34.1%	9.50e+07 23.2%	1.82e+11 36.2%
loop at sgn210_...	3.55e+11 26.0%	4.77e+09 33.4%	9.50e+07 23.2%	1.81e+11 36.0%
sgn210_3d.c:...	6.69e+10 4.9%	7.69e+08 5.4%	1.28e+07 3.1%	4.13e+10 8.2%
sgn210_3d.c:...	5.36e+10 3.9%	9.42e+08 6.6%	3.20e+06 0.8%	2.55e+10 5.1%

.51 FLOP per cycle

IWAVE Stencil - Why Low Performance?

- **Look at LLC misses to see demand fetch from memory**
- **Survey resource stalls from any source**

hpcviewer: asg.x

sgn210_3d.c

```
237
238     ratio_x[ix] = (1.0 - etaxdt) / (1.0 + etaxdt);
239     recip_x[ix] = 1.0 / (1.0 + etaxdt);
240       }
241
242     for ( iz = tid; iz < nz; iz += tsz )
243       {
244     etazdt = (*_epz) * dt2;
245
246     ratio_z = (1.0 - etazdt) / (1.0 + etazdt);
247     recip_z = 1.0 / (1.0 + etazdt);
248
249     for ( iy = 0; iy < ny; ++iy )
250       {
251         etaydt = (*_epy++) * dt2;
252
253         ratio_y = (1.0 - etaydt) / (1.0 + etaydt);
254         recip_y = 1.0 / (1.0 + etaydt);
255
256 #pragma ivdep
257         for ( _pxend = _px + nx; _px < _pxend; )
258           {
259         vx8 = _vx9[-1]; vx7 = _vx9[-2]; vx6 = _vx9[-3];
260         vx5 = _vx9[-4]; vx4 = _vx9[-5]; vx3 = _vx9[-6];
261         vx2 = _vx9[-7]; vx1 = _vx9[-8]; vx0 = _vx9[-9];
```

- LLC misses 3 orders of magnitude lower than cycles
- Resource stalls on par with total cycles

Calling Context View | Callers View | Flat View

Scope	PAPI_TOT_CYC:Sum (I)	MEM_LOAD_RETIRED:LLC_MISS:Sum (I)	RESOURCE_STALLS:ANY:Sum (I)
Experiment Aggregate Metrics	1.38e+12 100 %	1.04e+09 100 %	7.64e+11 100 %
main	1.38e+12 100.0	1.04e+09 100.0	7.64e+11 100.0
loop at asg.c: 133	1.38e+12 100.0	1.04e+09 100.0	7.64e+11 100.0
loop at asg.c: 239	1.38e+12 100.0	1.04e+09 100.0	7.64e+11 100.0
loop at asg.c: 239	1.38e+12 100.0	1.04e+09 100.0	7.64e+11 100.0
iwave_run	1.38e+12 100.0	1.04e+09 99.8%	7.64e+11 100.0
sgn_ts3d_210	8.83e+11 63.8%	6.74e+08 64.8%	4.82e+11 63.1%
sgn_ts3d_210p012	3.66e+11 26.5%	2.09e+08 20.1%	2.32e+11 30.4%
loop at sgn210_3d.c: 242	3.66e+11 26.4%	2.09e+08 20.1%	2.32e+11 30.4%
loop at sgn210_3d.c: 249	3.66e+11 26.4%	2.09e+08 20.1%	2.32e+11 30.3%
loop at sgn210_3d.c: 257	3.55e+11 25.7%	2.08e+08 20.0%	2.29e+11 30.0%
sgn210_3d.c: 268	6.68e+10 4.8%	1.64e+07 1.6%	4.69e+10 6.1%

IWAVE - Looking at Resource Stall Causes

2-sgn210_3d.c

```
250     {
251       etaydt = (*_epy++) * dt2;
252
253       ratio_y = (1.0 - etaydt) / (1.0 + etaydt);
254       recip_y = 1.0 / (1.0 + etaydt);
255
256 #pragma ivdep
257       for ( _pxend = _px + nx; _px < _pxend; )
258         {
259       vx8 = _vx9[-1]; vx7 = _vx9[-2]; vx6 = _vx9[-3];
260       vx5 = _vx9[-4]; vx4 = _vx9[-5]; vx3 = _vx9[-6];
261       vx2 = _vx9[-7]; vx1 = _vx9[-8]; vx0 = _vx9[-9];
262
263       vx9 = *_vx9++;
264
265       delta = ((vx9 - vx0) * C5x + (vx8 - vx1) * C4x + (vx7 - vx2) * C3x + (vx6 - vx3) * C2x + (vx5 - vx4) * C1x +
266            ((*_vy9++) - (*_vy0++)) * C5y + ((*_vy8++) - (*_vy1++)) * C4y + ((*_vy7++) - (*_vy2++)) * C3y +
267            ((*_vy6++) - (*_vy3++)) * C2y + ((*_vy5++) - (*_vy4++)) * C1y +
268            ((*_vz9++) - (*_vz0++)) * C5z + ((*_vz8++) - (*_vz1++)) * C4z + ((*_vz7++) - (*_vz2++)) * C3z +
269            ((*_vz6++) - (*_vz3++)) * C2z + ((*_vz5++) - (*_vz4++)) * C1z) * (*_mpx++);
270
271       (*_px) = (*_px) * (*ratio_x++) + delta * (*recip_x++);
272       _px++;
```

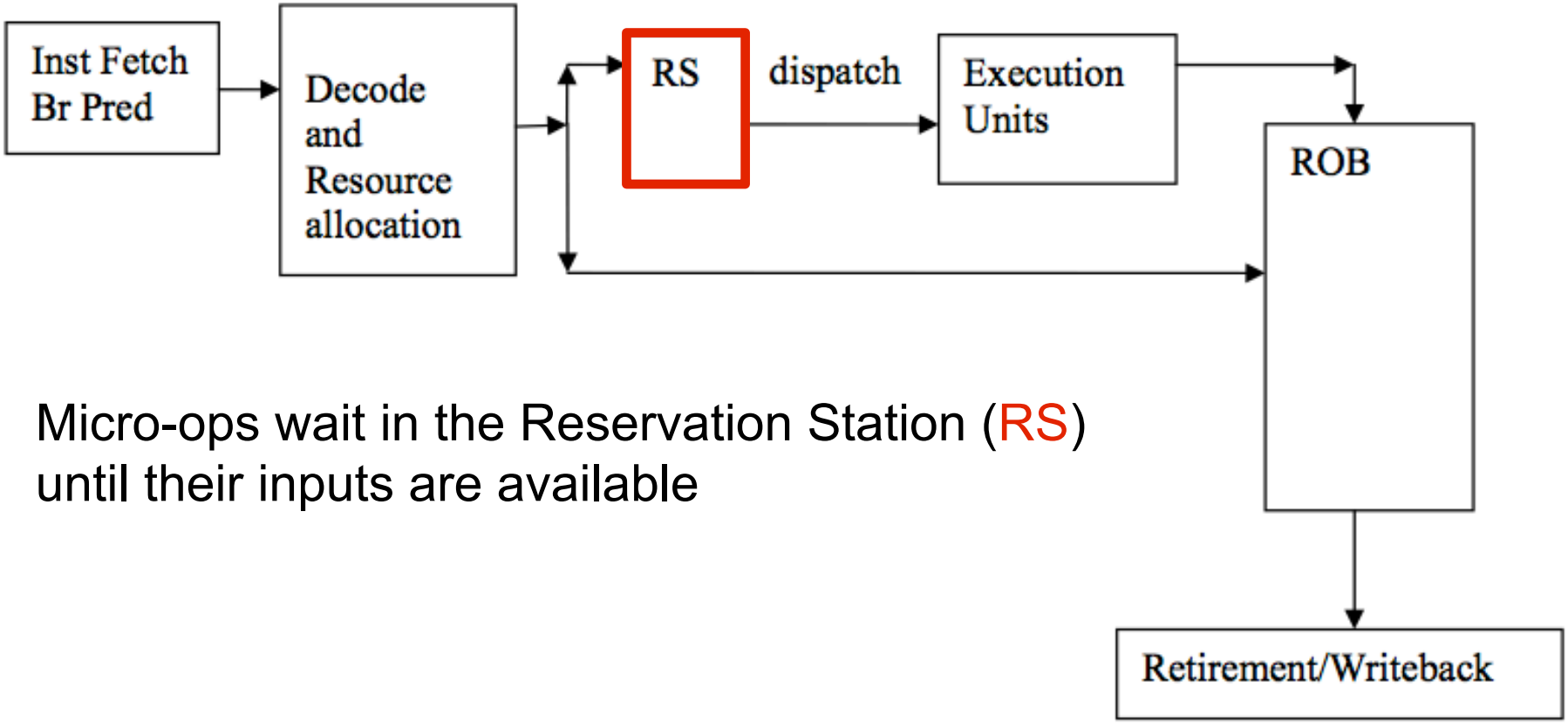
Dominant resource stalls
- LOADS
- Reservation station full

2-Calling Context View(asg.x) | 2-Callers View(asg.x) | 2-Flat View(asg.x)

Scope	RESOURCE_STALLS:LOAD:Sum (I)	RESOURCE_STALLS:STORE:Sum (I)	RESOURCE_STALLS:RS_FULL:Sum (I)	RESOURCE_STALLS:ROB_FULL:Sum (I)
Experiment Aggregate Metrics	1.76e+11 100 %	6.46e+10 100 %	3.81e+11 100 %	1.56e+11 100 %
main	1.76e+11 100.0	6.45e+10 100.0	3.81e+11 100.0	1.56e+11 100.0
loop at asg.c: 133	1.76e+11 100.0	6.45e+10 100.0	3.81e+11 100.0	1.56e+11 100.0
loop at asg.c: 239	1.76e+11 100.0	6.45e+10 100.0	3.81e+11 100.0	1.56e+11 100.0
loop at asg.c: 239	1.76e+11 100.0	6.45e+10 100.0	3.81e+11 100.0	1.56e+11 100.0
iwave_run	1.76e+11 100.0	6.45e+10 99.9%	3.81e+11 100.0	1.56e+11 99.9%
sgn_ts3d_210	1.75e+11 99.4%	3.16e+09 4.9%	2.17e+11 56.8%	1.14e+11 73.3%
sgn_ts3d_210p012	1.35e+11 76.7%	3.12e+09 4.8%	1.07e+11 28.0%	9.50e+07 0.1%
loop at sgn210_3d.c: 242	1.35e+11 76.7%	3.12e+09 4.8%	1.07e+11 28.0%	9.50e+07 0.1%
loop at sgn210_3d.c: 249	1.35e+11 76.7%	3.11e+09 4.8%	1.07e+11 28.0%	9.50e+07 0.1%
loop at sgn210_3d....	1.35e+11 76.6%	1.79e+09 2.8%	1.05e+11 27.6%	9.00e+07 0.1%
sgn210_3d.c: 268	3.69e+10 21.0%		1.45e+10 3.8%	4.00e+07 0.0%

Intel Nehalem Processor

A simplified view for understanding performance losses



Micro-ops wait in the Reservation Station (RS) until their inputs are available

Figure credit: David Levinthal

IWAVE - Looking at Memory System Usage

hpcviewer: asg.x

asg.c | sgn210_3d.c | iwave.c

```
249     for ( iy = 0; iy < ny; ++iy )
250       {
251         etaydt = (*_epy++) * dt2;
252
253         ratio_y = (1.0 - etaydt) / (1.0 + etaydt);
254         recip_y = 1.0 / (1.0 + etaydt);
255
256 #pragma ivdep
257         for ( _pxend = _px + nx; _px < _pxend; )
258           {
259         vx8 = _vx9[-1]; vx7 = _vx9[-2]; vx6 = _vx9[-3];
260         vx5 = _vx9[-4]; vx4 = _vx9[-5]; vx3 = _vx9[-6];
261         vx2 = _vx9[-7]; vx1 = _vx9[-8]; vx0 = _vx9[-9];
262
263         vx9 = *_vx9++;
264
265         delta = ((vx9 - vx0) * C5x + (vx8 - vx1) * C4x + (vx7 - vx2) * C3x + (vx6 - vx3) * C2x + (vx5 - vx4) * C1x +
266              ((*_vy9++) - (*_vy0++)) * C5y + ((*_vy8++) - (*_vy1++)) * C4y + ((*_vy7++) - (*_vy2++)) * C3y +
267              ((*_vy6++) - (*_vy3++)) * C2y + ((*_vy5++) - (*_vy4++)) * C1y +
268              ((*_vz9++) - (*_vz0++)) * C5z + ((*_vz8++) - (*_vz1++)) * C4z + ((*_vz7++) - (*_vz2++)) * C3z +
269              ((*_vz6++) - (*_vz3++)) * C2z + ((*_vz5++) - (*_vz4++)) * C1z) * (*_mpx++);
270
271         (*_px) = (*_px) * (*ratio_x++) + delta * (*recip_x++);
272         _px++;
273
274         (*_py) = (*_py) * ratio_y + delta * recip_y;
275         _py++;
276
277         (*_pz) = (*_pz) * ratio_z + delta * recip_z;
278         _pz++;
279           }
```

Analyze where the loads go

- L2 Hit - 1.51×10^{10}
- L3 Hit - 1.08×10^{9}
- Memory - 2.07×10^{8}

Calling Context View | Callers View | Flat View

Scope	PAPI_TOT_CYC:Sum (I)		MEM_LOAD_RETIRED:LLC_MISS:Sum (I)		MEM_LOAD_RETIRED:L2_HIT:Sum (I)		MEM_LOAD_RETIRED:LLC_UNSHARED_HIT:Sum (I)	
loop at asg.c: 239	1.85e+12	100.0	1.04e+09	100.0	4.64e+10	100 %	3.05e+09	100.0
loop at asg.c: 239	1.85e+12	100.0	1.04e+09	100.0	4.64e+10	100 %	3.05e+09	100.0
iwave_run	1.85e+12	99.9%	1.03e+09	99.8%	4.64e+10	100 %	3.05e+09	100.0
sgn_ts3d_210	8.86e+11	47.9%	6.66e+08	64.3%	1.92e+10	41.4%	2.82e+09	92.5%
sgn_ts3d_210p012	3.69e+11	19.9%	2.08e+08	20.1%	1.51e+10	32.4%	1.08e+09	35.4%
loop at sgn210_3d.c: 242	3.69e+11	19.9%	2.07e+08	20.0%	1.51e+10	32.4%	1.08e+09	35.4%
loop at sgn210_3d.c: 249	3.69e+11	19.9%	2.07e+08	20.0%	1.51e+10	32.4%	1.08e+09	35.4%
sgn210_3d.c: 244	6.00e+07	0.0%						
sgn210_3d.c: 247	3.40e+07	0.0%						
sgn210_3d.c: 246	4.00e+06	0.0%	4.00e+04	0.0%				
free	4.60e+07	0.0%	1.20e+05	0.0%				
loop at sgn210_3d.c: 152	2.00e+07	0.0%	8.00e+04	0.0%				
malloc	1.80e+07	0.0%	4.00e+04	0.0%				

Intel Nehalem Memory Hierarchy Costs

Intel® Xeon™ 5500 load Penalties

	L1D_HIT	Secondary Miss	L2 Hit	LLC Hit No Snoop	LLC Hit Clean Snoop	LLC Hit Snoop =HITM	Local Dram	Remote Dram	Remote Cache local home Fwd	Remote Cache Remote Home FWD	Remote Cache Local Home HITM	Remote Cache Remote home HITM
Mem_load_retired .L1d_hit	0 (By Def)											
Mem_load_retired .Hit_LFB		0->Max Val										
Mem_load_retired .L2_hit			6									
Mem_load_retired .LLC_Unshared_hit				~35								
Mem_load_retired .other_core_l2 _hit_hitm					~60	~75						
Mem_load_retired .LLC_Miss							~200	~350	~180	~180	~225 -250	~370
Mem_uncore_retired .Other_core_l2_hitm						~75						
Mem_uncore_retired .Local_Dram							~200				~225 -250	
Mem_uncore_retired .Remote_dram								~350				~370
Mem_uncore_retired. Remote_cache_ local_home_hit									~180			

Depend on frequency, dimms, bios, etc

Note: All latencies and memory access penalties shown are merely illustrative. Actual latencies will depend on (among other things) processor model, core and uncore frequencies, type, number and positioning of DIMMS, platform model, bios version and settings. Consult the platform manufacturer for optimal setting for any individual system. Then measure the actual properties of that system by running well established benchmarks.

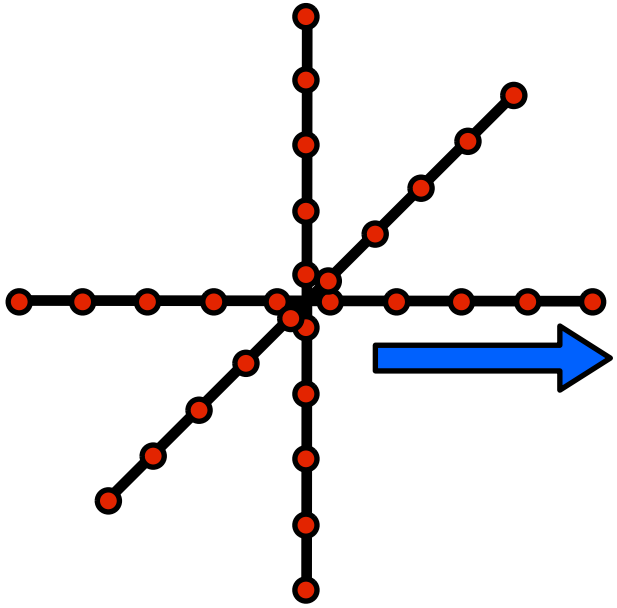
17

The Important Penalties Vary by a Factor of TEN

Figure Credit: David Levinthal

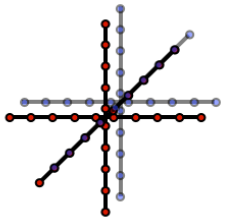
Principal Stencil Pattern

- **Execution under study**
  - **sgn_ts3d_210p012**
    - **10 points along each coordinate axis**
    - **sweep through memory along the X coordinate dimension**

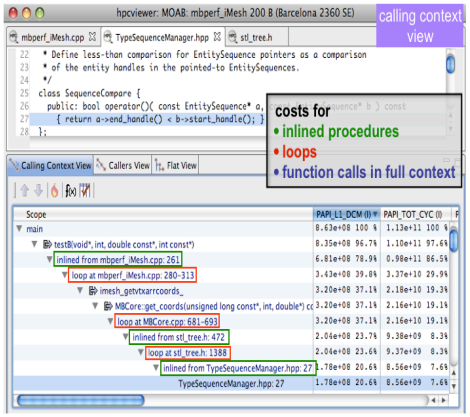


IWAVE Tuning Recommendations

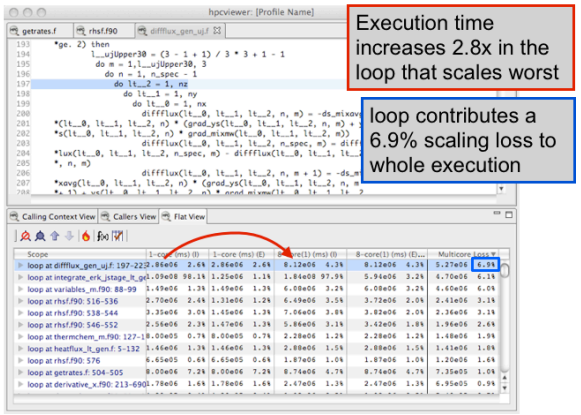
- **Computation vs. communication**
  - communication for the example studied is ~27% of iwave_run
  - no communication/computation overlap
  - compute on more data per core for higher parallel efficiency
- **I/O**
  - IWAVE uses serial Posix I/O for its input
  - using HDF5 and parallel I/O would be a higher performance choice
- **Stencil calculations**
  - IWAVE's stencil calculations are latency bound
    - spend most of their time waiting for data from L2 cache
  - need to make better use of the memory hierarchy
    - unrolling once in Y and Z coordinate dimensions will reuse data values immediately
      - currently, temporal reuse along Y and Z axis is long distance
      - unrolling in Y and Z: immediately reuse 9 of every 10 values loaded
    - pointer-based data access inhibits compiler-based tiling
      - tiling along Y and Z will be important for good cache reuse with large data



HPCToolkit Capabilities at a Glance

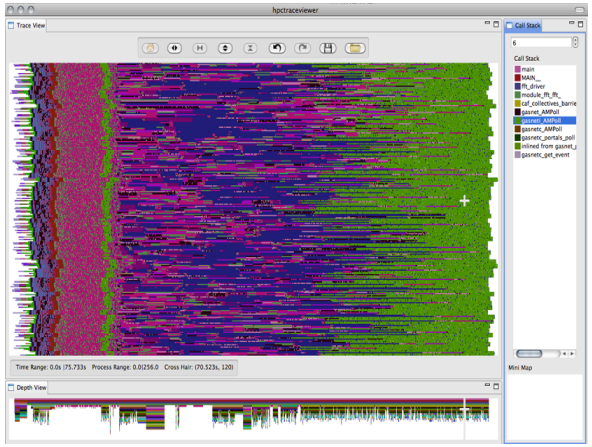


Attribute Costs to Code

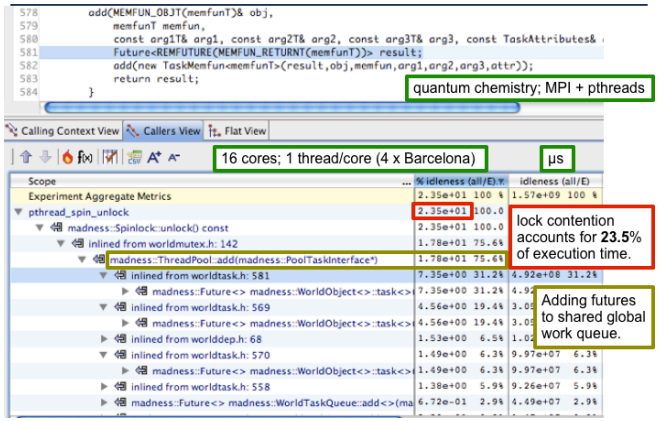


Pinpoint & Quantify Scaling Bottlenecks

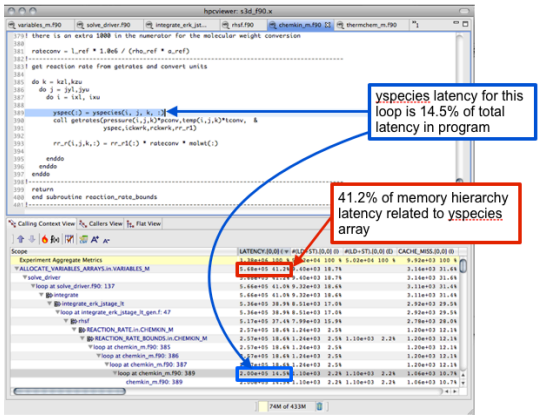
Assess Imbalance and Variability



Analyze Behavior over Time



Shift Blame from Symptoms to Causes



Associate Costs with Data

RICE

hpctoolkit.org

HPCToolkit Status

- **Operational today on**
  - 64- and 32-bit x86 systems running Linux (including Cray XT/E/K)
  - IBM Blue Gene/P
  - IBM Power7 systems running Linux
- **Emerging capabilities**
  - IBM Blue Gene/Q
  - NVIDIA GPU
    - measurement and reporting using GPU hardware counters
  - data centric analysis
- **Available as open source software at http://hpctoolkit.org**

Ongoing Work

- **Homogeneous nodes**
  - measurement and analysis for massive numbers of threads
  - "blame shifting" to pinpoint and quantify causes of idleness in OpenMP programs
- **Heterogeneous nodes**
  - "blame shifting" to pinpoint and quantify causes of CPU and GPU idleness in hybrid programs
  - derived metrics for GPU
- **Bandwidth monitoring of communication and I/O**
- **Future enhancements**
  - support for Intel MIC
  - provide higher-level prescriptive feedback

References

- **HPCToolkit Project: http://hpctoolkit.org**
- **David Levinthal. Performance Analysis Guide for Intel® CoreTM i7 Processor and Intel® XeonTM 5500 processors, Version 1.0, http://software.intel.com/sites/products/collateral/hpc/vtune/performance_analysis_guide.pdf**