

Profiling with HPCToolkit

Mark W. Krentel Department of Computer Science Rice University <u>krentel at rice.edu</u>

http://hpctoolkit.org



ALCF Computational Performance Workshop May 24, 2022

HPCToolkit Basic Features

- Run application natively (optimized) and every 100-200 times per second, interrupt program, unwind back to main(), record call stack, and combine these into a calling context tree (CCT).
- Combine sampling data with a static analysis of the program structure for loops and inline functions (hpcstruct).
- Present top-down, bottom-up and flat views of calling context tree (CCT) and time-sequence trace view. Metrics are displayed per source line in the context of their call path.
- Can sample on POSIX timers and Hardware Performance Counters (Perfmon or PAPI events): cycles, flops, cache misses, etc.
- Note: always include -g in compile flags (plus optimization) for attribution to source lines.

HPCToolkit Advanced Features

- Finely-tuned unwinder to handle multi-lingual, optimized code, no frame pointers, broken return pointers, stack trolling, etc.
- Derived metrics -- compute flops per cycle, or flops per memory reads, etc. and attribute to lines in source code.
- Compute strong and weak scaling loss, for example: strong: 8 * (time at 8K cores) - (time at 1K cores) weak: (time at 8K cores and 8x size) - (time at 1K cores)
- Load imbalance -- display distribution and variance in metrics across processes and threads.
- Blame shifting -- when thread is idle or waiting on a lock, blame the working threads or holder of lock.
- Inline sequences show full inline sequence for C++ templates.

New Features

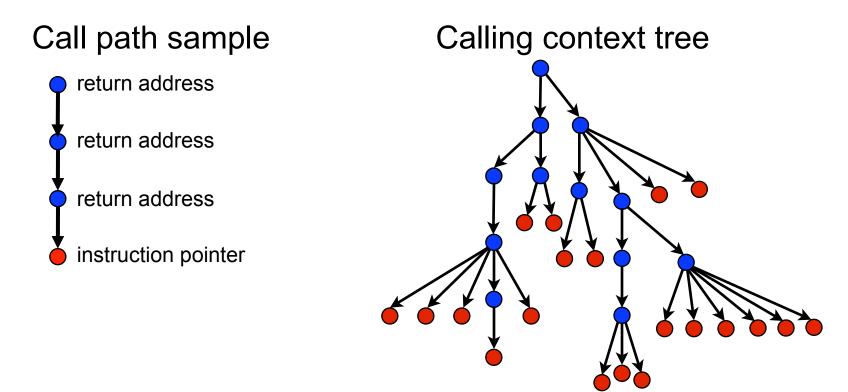
- Spack now build hpctoolkit and prereqs with spack and install with spack modules.
- Simplified use case for hpcstruct and hpcprof.
- Kernel Blocktime use Perf Events to count time spent blocked inside kernel, eg, I/O, barriers, locks, etc. (requires kernel perf events paranoid level 0 or 1).

— hpcrun -e CYCLES -e BLOCKTIME ...

- GPU full support for Nvidia and AMD, in progress for Intel.
- Support for OpenMP parallel regions splice thread call paths onto master thread and identify work and idle (requires libomp replacement library), part of OpenMP 5.

Call Path Profiling

Measure and attribute costs in context sample timer or hardware counter overflows gather calling context using stack unwinding



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

Where to find HPCToolkit

 Home site: user's manual, build instructions, links to source code, download viewers. http://hpctoolkit.org/

 On theta, available as module hpctoolkit (includes hpcviewer on theta login nodes). module load hpctoolkit/2022.05.15 (theta) module load hpctoolkit/2022.05.15-gpu (theta-gpu)

See: /soft/perftools/hpctoolkit/workshop-2022 for build/run notes, example databases, etc.

- Source code on GitHub
 <u>https://github.com/hpctoolkit</u>
 git clone <u>https://github.com/hpctoolkit/hpctoolkit</u>
 spack build instructions: README.Install
- Send questions to:

hpctoolkit-forum at mailman.rice.edu

Quickstart for theta-gpu

• On theta-gpu,

- module load hpctoolkit/2022.05.15-gpu

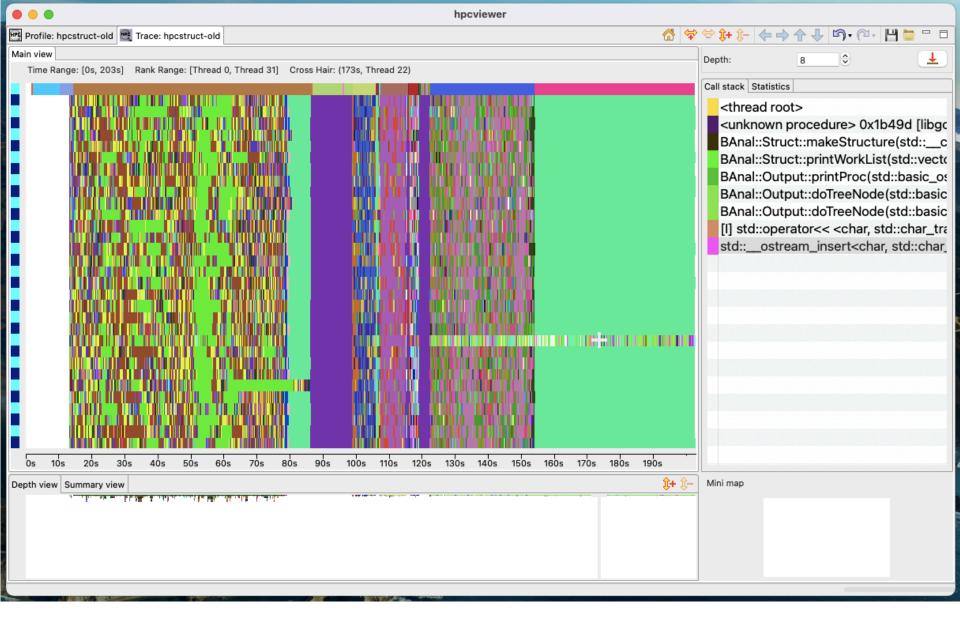
- Run application as follows. The first example is lowoverhead. The 'pc' option displays the internals of the gpu kernels but can cause high overhead (1.5x to 4x).
 - hpcrun [-t] -e REALTIME -e gpu=nvidia app ...
 - hpcrun [-t] -e REALTIME -e gpu=nvidia,pc app …
- Post-run analysis.
 - hpcstruct hpctoolkit-measurements-directory
 - hpcprof hpctoolkit-measurements-directory
- Finally, run hpcviewer and select database directory in the File menu chooser.

Using OpenMP Tools Library

- Use hpctoolkit ompt module.
 module load hpctoolkit/2020.04.ompt
- Compile with -fopenmp, but on hpclink link line, replace -fopenmp with libomp.a from LLVM runtime. Supports GNU, Intel and Clang. On theta, /projects/Tools/hpctoolkit/pkgs-theta/llvm-openmp/lib/

libomp.a

- Add event OMP_IDLE (no number) plus time-based event: REALTIME, PAPI_TOT_CYC or CYCLES.
- Workarounds on theta to turn off thread affinity. aprun —cc none ... export KMP_AFFINITY=none



Trace of hpcstruct analyzing 8 Gig .so file.

🔄 Fronie, da 🔤 Trace, da 🔄 Fronie, da 🔤 Trace, da main.cc

main.cc

Metric properties

CycleTracking.cc

CollisionEvent.co	: 🖾	MacroscopicCrossSection.cc
-------------------	-----	----------------------------

66 67 68	for (int isoIndex = 0; isoIndex < numIsos && currentCrossSection >= 0; isoIndex++) {
69	<pre>int uniqueNumber = monteCarlo->_materialDatabase->_mat[globalMatIndex]iso[isoIndex]gid;</pre>
70	int numReacts = monteCarlo->_nuclearData->getNumberReactions(uniqueNumber);
71	<pre>for (int reactIndex = 0; reactIndex < numReacts; reactIndex++)</pre>
72	{
73	<pre>currentCrossSection -= macroscopicCrossSection(monteCarlo, reactIndex, mc_particle.domain,</pre>
74	<pre>isoIndex, mc_particle.energy_group);</pre>
75	if (currentCrossSection < 0)
76	{
77	<pre>selectedIso = isoIndex;</pre>
78	<pre>selectedUniqueNumber = uniqueNumber;</pre>

Top-down view Bottom-up view Flat view

🕆 🕂 👌 🌆 🐺 🕅 🗛 🚛 💷 😵

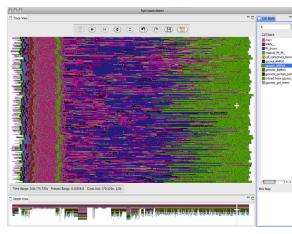
Scope	GINS:Sum (I)	REAL
 127 ⇒device_stubZ19CycleTrackingKernelP10MonteCarloiP13ParticleVaultS2_(MonteCarlo*, int, P 	9.37e+10	100.0%
 · 14 ⇒ [I] cudaLaunchKernel<char></char> 	9.37e+10	100.0%
4 211 ⇒ <gpu kernel=""></gpu>	9.37e+10	100.0%
✓ ⇒ CycleTrackingKernel(MonteCarlo*, int, ParticleVault*, ParticleVault*) [54b67c8fdf4def7a08c8d1fe1	9.37e+10	100.0%
 132 ⇒ CycleTrackingGuts(MonteCarlo*, int, ParticleVault*, ParticleVault*) [54b67c8fdf4def7a08c8d 	9.36e+10	100.0%
 26 ⇒ [I] CycleTrackingFunction(MonteCarlo*, MC_Particle&, int, ParticleVault*, ParticleVault*) 	7.00e+10	74.7%
Ioop at CycleTracking.cc: 118	7.00e+10	74.7%
 G3 ⇒ CollisionEvent(MonteCarlo*, MC_Particle&, unsigned int) [54b67c8fdf4def7a08c8d1fe1b0 	4.41e+10	47.0%
 loop at CollisionEvent.cc: 67 	3.40e+10	36.3%
 loop at CollisionEvent.cc: 71 	3.21e+10	34.2%
 - 73 ⇒ macroscopicCrossSection(MonteCarlo*, int, int, int, int, int) [54b67c8fdf4def7a08c8d1f 	2.98e+10	31.8%
 41 ⇒ NuclearData::getReactionCrossSection(unsigned int, unsigned int, unsigned int) [54b6 	1.74e+10	18.6%
▶ 253 ⇒ III NuclearDataReaction::getCrossSection(unsigned int)	5.85e+09	6.2%

Code inside GPU kernel in quicksilver proxy app.

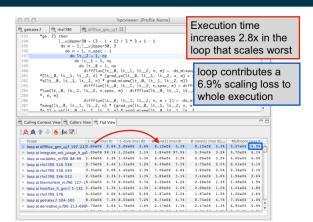
HPCToolkit Capabilities at a Glance

22 * Define less-than comparison for EntitySequence point		
23 • of the entity handles in the pointed-to EntitySequent 24 */ 25 class SequenceCompare {		0
<pre>26 public: bool operator()(const EntitySequence* a, 27 { return a->end_handle() < b->start_handle(); } 28 };</pre>	osts for server be inlined procedu	
College Contractions in College Man in Statistics	loops function calls ir	
] ☆ ♣ <mark>6</mark> ∰ ₩ ₩		Tun context
Scope	PAPI_L1_DCM (I) #	
▼ main	8.63e+08 100 %	1.13e+11 100 %
test8(void*, int, double const*, int const*)	8.35e+08 96.7%	
inlined from mbperf_iMesh.cpp: 261	6.81e+08 78.9%	
Ioop at mbperf_iMesh.cpp: 280-313	3.43e+08 39.8%	3.37e+10 29.9%
Imesh_getvtxarrcoords_	3.20e+08 37.1%	2.18e+10 19.3%
MBCore::get_coords(unsigned long const*, int, do	uble*) cc 3.20e+08 37.1%	2.16e+10 19.1%
Ioop 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%
TypeSeguenceManager.hpp: 27	1.78e+08 20.68	8.56e+09 7.6%

Attribute Costs to Code



Analyze Behavior over Time



Pinpoint & Quantify **Scaling Bottlenecks**

Future<REMFUTURE(MEMFUN_RETURNT(memfunT))> result;

madness::ThreadPool::add(madness::PoolTaskInterface*

madness::Future<> madness::WorldObject<>::tas

inlined from worldtask.h: 581

inlined from worldtask.h: 569

Inlined from worlddep.h: 68

🔻 🗐 inlined from worldtask.h: 570

Inlined from worldtask.h: 558

const arg1T& arg1, const arg2T& arg2, const arg3T& arg3, const TaskAttributes&

16 cores: 1 thread/core (4 x Barcelona)

▶ 📾 madness::Future<> madness::WorldObject<>::task<> 4.56e+00 19.4% 3.05

Shift Blame from

Symptoms to Causes

▶ 🛱 madness::Future<> madness::WorldObject<>::task<> (1.49e+00 6.3% 9.97e+07 6.3%

▶ 🗐 madness::Future<> madness::WorldTaskQueue::add<>(ma 6.72e-01 2.9% 4.49e+07 2.9%

μs

1.57e+09 100 %

lock contention

of execution time.

% idleness (all/E). . idleness (all/E)

7.350+00 31.29 4.920+08 31.29

1.53e+00 6.5% 1.02 work queue.

.49e+00 6.3% 9.97e+07 6.3

1.38e+00 5.9% 9.26e+07 5.9%

2.35e+01 100 %

2.35e+01 100.

2.35e+01 100.0

1.78e+01 75.6%

1.78e+01 75.6%

7.35e+00 31.2% 4.92

4.560+00 19.48 3.0

add(new TaskMemfun<memfunT>(result,obj,memfun,arg1,arg2,arg3,attr));

add(MEMEUN_OBJT(memfunT)& obj.

memfunT memfun,

return result:

😵 Calling Context View 🛝 Callers View 🏦 Flat View

madness::Spinlock:unlock() const

inlined from worldmutex.h: 142

🕆 🕂 🗄 🕅 🕅 🐨 🛧 🛧

Experiment Aggregate Metrics

othread spin unlock

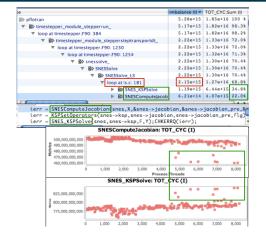
580

581

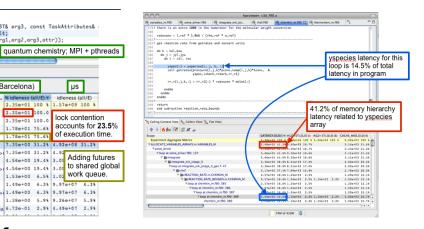
582

583

Scope



Assess Imbalance and Variability



Associate Costs with Data

hpctoolkit.org

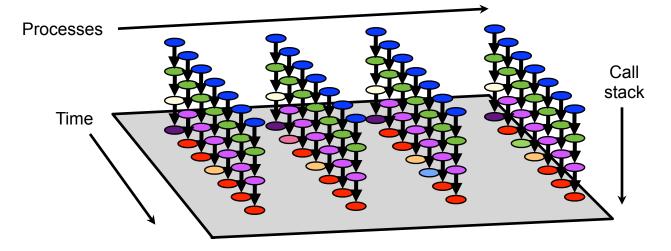
Understanding Temporal Behavior

- Profiling compresses out the temporal dimension
 —temporal patterns, e.g. serialization, are invisible in profiles
- What can we do? Trace call path samples

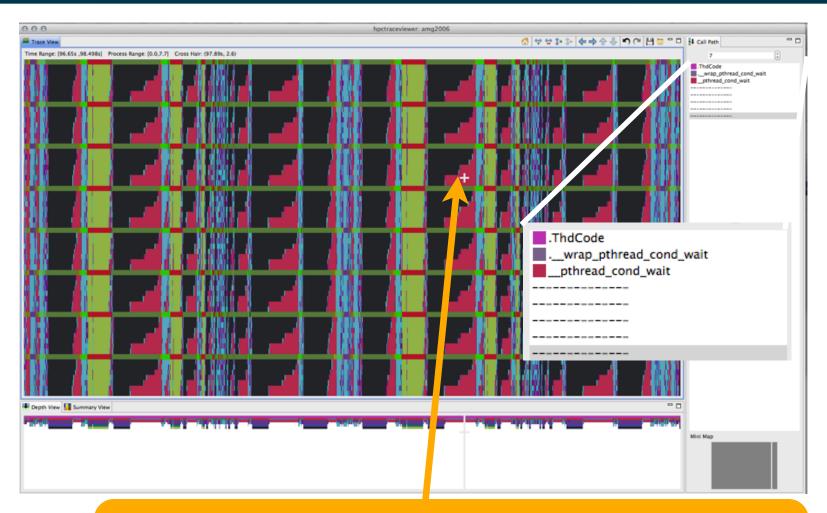
-sketch:

- 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



AMG2006: 8PE x 8 OMP Threads



OpenMP loop in hypre_BoomerAMGRelax using static scheduling has load imbalance; threads idle for a significant fraction of their time

Code-centric view: hypre_BoomerAMGRelax

000	hpcviewer: amg2006			
par_relax.c 🛛				- [
1632 #define HYPRE_SMP_PRIVATE i				
1633 #include "/utilities/hypre_smp_for	loop.h"			
<pre>1634 for (i = 0; i < n; i++)</pre>				
<pre>1635 tmp_data[i] = u_data[i];</pre>				
1636 #define HYPRE_SMP_PRIVATE i, ii, j, jj,	ns,ne,res,rest,size	Note: The high	lighted Open	MD loop in
1637 #include "/utilities/hypre_smp_for	loop.h"			
<pre>1638 for (j = 0; j < num_threa</pre>	ads; j++)	hypre_Boomer/		
1639 {		only 4.6% of the		
<pre>1640 size = n/num_threads;</pre>		benchmark ru	in. In real runs	s, solves
<pre>1641 rest = n - size*num_threads;</pre>		using this loo	p are a domin	ant cost
1642 if (j < rest)		J J	•	
1643 {				
<pre>1644 ns = j*size+j;</pre>				_
<pre>1645 ne = (j+1)*size+j+1;</pre>		across all inst	ancos of this	
1646 }				
1647 else		loop in hypre	_boomeraw	IGReiax
1648 {			-	
1648 { 1649 ns = j*size+rest;		19.7% of time	-	
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest;		19.7% of time	in this loop	is spent
<pre>1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 }</pre>			in this loop	is spent
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest;	x / * intenien anista finat	19.7% of time	in this loop	is spent
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 }	\ /* istanian asiata fiast	19.7% of time	in this loop	is spent this loop
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1662 Calling Context View ### Flat View 🔀	* /* istanism saista fisst	19.7% of time	in this loop	is spent
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } Calling Context View ### Flat View 🕱	\ /* istanism saists first	19.7% of time	in this loop	is spent this loop
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1651 } 1652 1651 1652 1651 1652 1651 1652 1653 1654 1655 1655 1655 1655 1655 1655 1655 1655 1655 1655 1655 1655 1650 1651 1652 1653 1654 1655 1655 1656 1657 1658	WALLCLOCK (us):Sum (l) v	19.7% of time idle idle w.r.t. t	in this loop	is spent this loop
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1652 Calling Context View ↑ Flat View 2 Calling Context View ↑ Flat View 2 A* A-		19.7% of time idle idle w.r.t. t	e in this loop otal effort in	is spent this loop
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1651 } 1652 Image: Context View 1653 } 1654 Image: Context View 1655 J 1651 J 1652 Image: Context View 1653 J 1654 J 1655 J 1655 J 1655 J 1655 J 1655 J 1657 J 1658 J 1659 J 1650 J 1651 J 1650 J 1651 J 1651 J 1651 J 1651 J 1651 J 1651 J 1652 J 1653 J 1644 J 1655 J 1650	WALLCLOCK (us):Sum (I) v	19.7% of time idle idle w.r.t. t	e in this loop otal effort in idleness %	is spent this loop
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1651 } 1651 } 1651 } 1652 Image: Context View 1651 } 1651 } 1652 Image: Context View 1653 Image: Context View 1654 Image: Context View 1655 Image: Context View 1657 Image: Context View 1658 Image: Context View 1659 Image: Context View 1650 Image: Context View 1651 Image: Context View 1651 Image: Context View 1651 Image: Context View 1651 Image: Context View 1652 Image: Context View 1653 Image: Context View 1654 Image: Context View 1655 Image: Context View 1656 Image: Context View 1657 Image: Context View 1658 Image: Context View	WALLCLOCK (us):Sum (l) 6.81e+08 11.18	19.7% of time idle idle w.r.t. t	in this loop total effort in idleness % 7.97e+01 7.97e+01 7.97e+01 7.97e+01	work % 2.03e+01 2.03e+01 2.03e+01
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1651 } 1652 Image: Context View 1653 } 1654 Image: Context View 1655 J 1651 J 1652 Image: Context View 1653 J 1654 J 1655 J 1655 J 1655 J 1655 J 1655 J 1657 J 1658 J 1659 J 1650 J 1651 J 1650 J 1651 J 1651 J 1651 J 1651 J 1651 J 1651 J 1652 J 1653 J 1644 J 1655 J 1650	WALLCLOCK (us):Sum (l) 6.81e+08 11.1% 6.81e+08 11.1%	19.7% of time idle idle w.r.t. t	in this loop total effort in idleness % 7.97e+01 7.97e+01 7.97e+01 7.97e+01	work % 2.03e+01 2.03e+01
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1651 } 1651 } 1651 } 1652 Image: Context View 1651 } 1651 } 1652 Image: Context View 1653 Image: Context View 1654 Image: Context View 1655 Image: Context View 1657 Image: Context View 1658 Image: Context View 1659 Image: Context View 1650 Image: Context View 1651 Image: Context View 1651 Image: Context View 1651 Image: Context View 1651 Image: Context View 1652 Image: Context View 1653 Image: Context View 1654 Image: Context View 1655 Image: Context View 1656 Image: Context View 1657 Image: Context View 1658 Image: Context View	WALLCLOCK (us):Sum (l) 6.81e+08 11.18 6.81e+08 11.18 6.81e+08 11.18	19.7% of time idle idle w.r.t. t WALLCLOCK (us):Sum (E) 3.20e+04 0.0 1.44e+06 0.0	in this loop otal effort in idleness % 7.97e+01 7.97e+01 7.97e+01 7.97e+01 8 2.35e+01	work % 2.03e+01 2.03e+01 2.03e+01
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1651 } 1651 } 1651 } 1652 Image: Context View Image: Calling Context View Image: Flat View 1651 } 1651 } 1651 } 1651 } 1652 Image: Context View Image: Calling Context View Image: Flat View 1653 Image: Calling Context View Image: Calling Context View Image: Flat View Imag	WALLCLOCK (us):Sum (l) ▼ 6.81e+08 11.1% 6.81e+08 11.1% 6.81e+08 11.1% 3.77e+08 6.1%	19.7% of time idle idle w.r.t. t WALLCLOCK (us):Sum (E) 3.20e+04 0.0 1.44e+06 0.0	in this loop otal effort in idleness % 7.97e+01 7.97e+01 7.97e+01 8 2.35e+01 8 4.80e+01	is spent this loop © E work % 2.03e+01 2.03e+01 2.03e+01 7.65e+01
1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1651 } 1651 } 1651 } 1652 Image: Context View 1651 } 1651 } 1652 Image: Context View 1653 Image: Context View 1654 Image: Context View 1655 Image: Context View 1657 Image: Context View 1658 Image: Context View 1659 Image: Context View 1650 Image: Context View 160 Image: Context View	WALLCLOCK (us):Sum (l) ▼ 6.81e+08 11.1% 6.81e+08 11.1% 6.81e+08 11.1% 3.77e+08 6.1% 3.16e+08 5.2%	19.7% of time idle idle w.r.t. t WALLCLOCK (us):Sum (E) 3.20e+04 0.0 1.44e+06 0.0 1.00e+03 0.0	in this loop otal effort in idleness % 7.97e+01 7.97e+01 7.97e+01 8 2.35e+01 8 8.75e+01	is spent this loop work % 2.03e+01 2.03e+01 2.03e+01 7.65e+01 5.20e+01 1.25e+01
<pre>1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1000 Calling Context View T. Flat View A* A* Scope hypre PCGSetup hypre BoomerAMGSetup hypre BoomerAMGSetup </pre>	WALLCLOCK (us):Sum (l) 6.81e+08 11.1% 6.81e+08 11.1% 6.81e+08 11.1% 3.77e+08 6.1% 3.16e+08 5.2% 3.01e+08 4.9%	19.7% of time idle idle w.r.t. t WALLCLOCK (us):Sum (E) 3.20e+04 0.0 1.44e+06 0.0 1.00e+03 0.0 2.81e+08 4.6	in this loop sotal effort in idleness % 7.97e+01 7.97e+01 7.97e+01 8 2.35e+01 8 4.80e+01 8 8.75e+01 8 1.97e+01	is spent this loop
<pre>1648 { 1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1000 Calling Context View T. Flat View A* A* Scope Nypre PCGSetup Nypre BoomerAMGSetup Nypre BoomerAMGSe</pre>	WALLCLOCK (us):Sum (l) 6.81e+08 11.1% 6.81e+08 11.1% 6.81e+08 11.1% 3.77e+08 6.1% 3.16e+08 5.2% 3.01e+08 4.9% 2.81e+08 4.6%	19.7% of time idle idle w.r.t. t WALLCLOCK (us):Sum (E) 3.20e+04 0.0 1.44e+06 0.0 1.00e+03 0.0 2.81e+08 4.6 2.00e+03 0.0	in this loop otal effort in idleness % 7.97e+01 7.97e+01 7.97e+01 8 2.35e+01 8 4.80e+01 8 8.75e+01 8 1.97e+01 8 1.97e+01	is spent this loop

Serial Code in AMG2006 8 PE, 8 Threads

000	hpcviewer: amg2006	
par_relax.c 🕱		- 6
1632 #define HYPRE_SMP_PRIVATE i		
1633 #include "/utilities/hypre_sm	n forloon h"	
1634 for (i = 0; i < n; i		
1635 tmp_data[i] = u_data[-	
1636 #define HYPRE_SMP_PRIVATE i,ii,		
1637 #include "/utilities/hypre_sm		
1638 for (j = 0; j < num_		
1639 {		
1640 size = n/num_threads;		
1641 rest = n - size*num_thr	eads:	7 worker threads are idle
1642 if (j < rest)	,	
1643 {		in each process while its
<pre>1644 ns = j*size+j;</pre>		main MPI thread is
<pre>1645 ne = (j+1)*size+j+1;</pre>		
1646 }		working
1647 else		
1648 {		
•		
<pre>1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest 1651 }</pre>		
<pre>1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest 1651 }</pre>	:; 	
1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest 1651 }		
<pre>1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest 1651 } 1651 }</pre>		, — E
1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest 1651 } Calling Context View ∰, Flat View ⊠		
ns = j*size+rest; $ne = (j+1)*size+rest$ $rest$		- [
ns = j*size+rest; $ne = (j+1)*size+rest$ $(calling Context View ft. Flat View S)$ $(calling Context View ft. Flat View S)$	/• intenien neinte finst •/	- [
1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1652 1651 1651 1651 1651 1652 1653 1654 1655 1651 1652 1653 1654 1655 1657 1658 1658 1659 1650	WALLCLOCK (us):Sum (l)	CLLCLOCK (us):Sum (E) idleness % work %
$\begin{array}{c} ns = j*size+rest;\\ ne = (j+1)*size+rest\\ 1651 \\ \end{array}$ $\begin{array}{c} Calling Context View \\ \hline f \\ \hline \hline f \\ \hline f \\ \hline \hline \hline f \\ \hline \hline \hline \hline$	WALLCLOCK (us):Sum (l) WA	C E ALLCLOCK (us):Sum (E) idleness % work % 6.13e+09 100 % 4.91e+01 5.09e+01
ns = j*size+rest; ne = (j+1)*size+rest; 1650 1651 3 Calling Context View f_{\pm} . Flat View \Im Calling	WALLCLOCK (us):Sum (l) W4 6.13e+09 100 % 3.64e+07 0.6%	CLLCLOCK (us):Sum (E) idleness % work % 6.13e+09 100 % 4.91e+01 5.09e+01 3.64e+07 0.6% 8.74e+01 1.26e+01
$\begin{array}{c} ns = j*size+rest;\\ ne = (j+1)*size+rest;\\ l650 \\ ne = (j+1)*size+rest;\\ l651 \\ \end{array}$ $\begin{array}{c} calling Context View \\ \hline \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$	WALLCLOCK (us):Sum (l) W4 6.13e+09 100 % 3.64e+07 0.6% 8.47e+06 0.1%	LLCLOCK (us):Sum (E) idleness % work % 6.13e+09 100 % 4.91e+01 5.09e+01 3.64e+07 0.6% 8.74e+01 1.26e+01 8.47e+06 0.1% 8.75e+01 1.25e+01
ns = j*size+rest; ne = (j+1)*size+rest; ne = (j+1)*size+rest; calling Context View \ddagger ; Flat View \bowtie calling Context View \ddagger ; Flat View \bowtie calling Context View \ddagger ; Flat View \bowtie calling Context View \ddagger ; Flat View \bowtie \bigstar \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit \clubsuit	WALLCLOCK (us):Sum (l) W4 6.13e+09 100 % 3.64e+07 0.6% 8.47e+06 0.1% 7.80e+06 0.1%	LLCLOCK (us):Sum (E) idleness % work % 6.13e+09 100 % 4.91e+01 5.09e+01 3.64e+07 0.6% 8.74e+01 1.26e+01 8.47e+06 0.1% 8.75e+01 1.25e+01 7.80e+06 0.1% 8.75e+01 1.25e+01
ns = j*size+rest; ne = (j+1)*size+rest; ne = (j+1)*size+rest; calling Context View f_{f**} Flat View \Im calling Context View f_{f**} Flat View \Im $f \cong$ A^* A- cope Experiment Aggregate Metrics loop at binsearch.c: 78 loop at amg linklist.c: 78 loop at amg linklist.c: 226 inlined from RecChannel.h: 349 inlined from InjGroup.h: 191	WALLCLOCK (us):Sum (l) WA 6.13e+09 100 % 3.64e+07 0.6% 8.47e+06 0.1% 7.80e+06 0.1% 7.91e+06 0.1%	LLCLOCK (us):Sum (E) idleness % work % 6.13e+09 100 % 4.91e+01 5.09e+01 3.64e+07 0.6% 8.74e+01 1.26e+01 8.47e+06 0.1% 8.75e+01 1.25e+01 7.80e+06 0.1% 8.75e+01 1.25e+01 7.48e+06 0.1% 8.68e+01 1.32e+01 3.38e+06 0.1% 8.69e+01 1.31e+01 2.89e+06 0.0% 8.69e+01 1.31e+01
1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1651 } 1651 } 1652 1651 } 1651 } 1652 1651 } 1652 1653 1651 1652 1653 1654 1655 1655 1655 1655 1655 1655 1655 1657 1658 1659 1650 1651 1652 1653 1654 1655 1657 1658 1659 1600	WALLCLOCK (us):Sum (l)	LLCLOCK (us):Sum (E) idleness % work % 6.13e+09 100 % 4.91e+01 5.09e+01 3.64e+07 0.6% 8.74e+01 1.26e+01 8.47e+06 0.1% 8.75e+01 1.25e+01 7.80e+06 0.1% 8.75e+01 1.25e+01 7.48e+06 0.1% 8.68e+01 1.32e+01 3.38e+06 0.1% 8.69e+01 1.31e+01
1649 ns = j*size+rest; 1650 ne = (j+1)*size+rest; 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 } 1651 > 1651 > 1651 > 1651 > 1651 > 1651 > 1651 > 1652 1653 > 1654 > 1655 1657 > 1658 1659 amg 1690 at amg 1600 at amg 1600 at amg 1610 from RecChannel.h: 349 16110 from Fifo.h: 195	WALLCLOCK (us):Sum (l) WA 6.13e+09 100 % 3.64e+07 0.6% 8.47e+06 0.1% 7.80e+06 0.1% 7.91e+06 0.1% 3.42e+06 0.1% 2.89e+06 0.0%	LLCLOCK (us):Sum (E) idleness % work % 6.13e+09 100 % 4.91e+01 5.09e+01 3.64e+07 0.6% 8.74e+01 1.26e+01 8.47e+06 0.1% 8.75e+01 1.25e+01 7.80e+06 0.1% 8.75e+01 1.25e+01 7.48e+06 0.1% 8.68e+01 1.32e+01 3.38e+06 0.1% 8.69e+01 1.31e+01 2.89e+06 0.0% 8.69e+01 1.31e+01

Pinpointing and Quantifying Scalability Bottlenecks

