Integrating Analysis and Computation:

Experiences with In-Situ and In-Transit Approaches

Approved for Public Release: SAND2013-2302C

Ron A. Oldfield, David Rogers,
Ken Moreland, and Nathan Fabian

Sandia National Laboratories
Albuquerque, NM, USA

SOS 17 Conference
Jeckyll Island, GA
March 2013

Sandia National Laboratories

Exceptional service in the national interest

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Some I/O Issues for Exascale

- Storage systems are the slowest, most fragile, part of an HPC system
  - Scaling to extreme client counts is challenging
  - POSIX semantics gets in the way, …
- Current usage models not appropriate for Petascale, much less Exascale
  - Checkpoints are a ***HUGE*** concern for I/O…currently primary focus of FS
  - App workflow uses storage as a communication conduit
    - Simulate, ***store***, analyze, ***store***, refine, ***store***, … most of the data is transient
- One way to relieve I/O pressure on the FS is to integrate simulation and analysis
  1. ***In-situ processing*** provides "tightly coupled" analysis through libraries linked directly with the simulation.
  2. ***In-transit processing*** provides "loosely coupled" analysis by performing analysis on separate processing resources.

This talk presents a detailed comparison of these approaches.

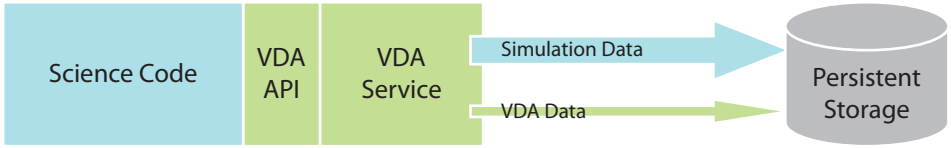
Pros/Cons for In Situ and In Transit

Sandia National Laboratories

- In situ
  - + Most common approach for integrating analysis
  - + Straight forward to use (just function calls)
  - + If implemented right, could reuse application data structures
  - − Synchronous (app must wait for viz to complete)
  - − May add significant memory, computation, comm requirements
  - − May cause concerns for stability, scalability, resilience.
- In transit
  - + Minimal client overhead (addresses resilience, scalability, …)
  - + Asynchronous (overlap computation and analysis)
  - + Analysis can execute in different environment (e.g., linux vs lwk)
  - − Requires additional compute resources
  - − New use case: more complicated to schedule, load balance, …

Our *In situ* and *in transit* workflows

- Our ***In situ*** workflows uses *Catalyst*, an open source, VTK-based analysis library derived from ParaView.



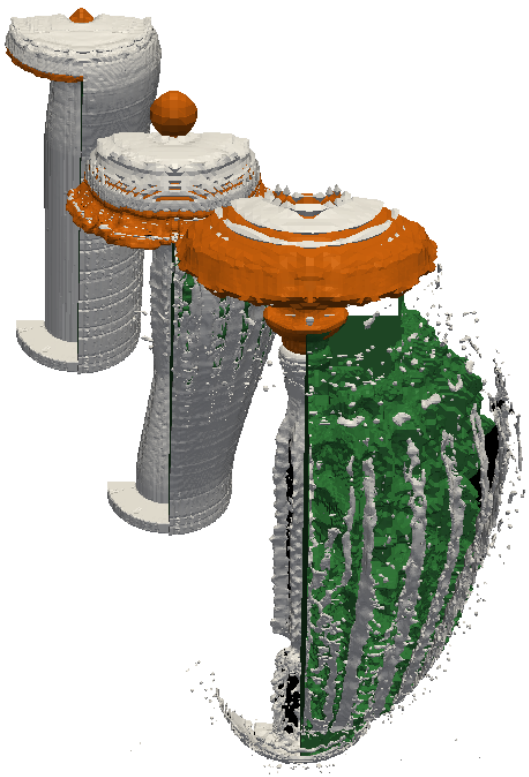
- Our ***In transit*** workflows use the Network Scalable Service Interface (*Nessie*) to communicate with analysis services allocated on separate compute resources. Nessie is an open source data services library that is part of the Trilinos I/O Support package.

The science code uses the same API for both approaches, making comparison between the two approaches trivial.

Customer Driven Use Case

Characterize fragments in an explosion simulation

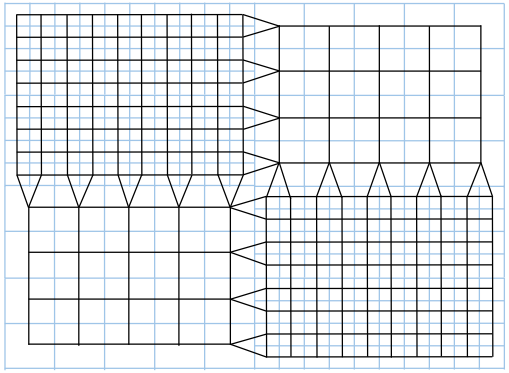
- An analysis step critical for understanding shock physics
- Partner: Jason Wilke – SNL Analyst
- Critical steps
  - Fragment detection (multiple operations required)
  - Characterize fragments (mass, velocity, etc.)
  - Extract useful information



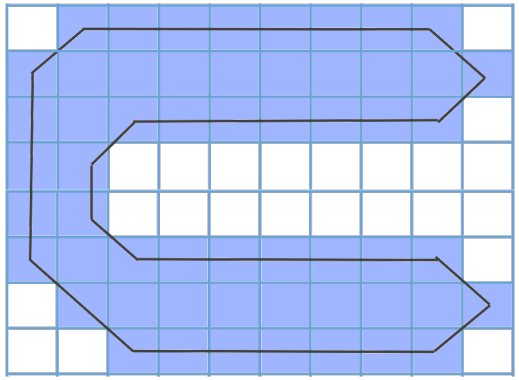
Experiments focused on identifying the fragments. This operation is a significantly complex part the analysis, so it serves as a useful way to characterize the operations in the driver use case.

Fragment detection

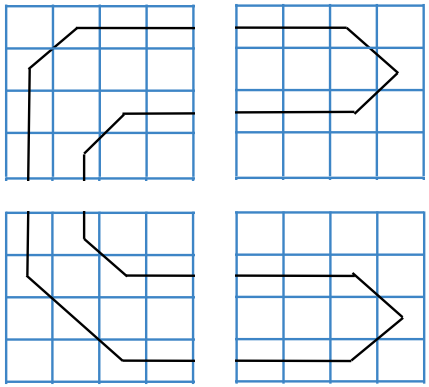
- Operations required for fragment detection (requires a watertight surface)
  1. Find block neighbors
  2. Build a conforming mesh over AMR boundaries
  3. Identify boundaries of fragments
  4. Find fragment components that are connected *(not in these experiments)*



Step 2



Step 3

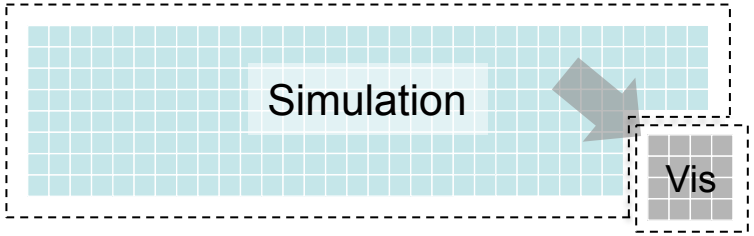


Step 4

Application Workflows

1. ***In situ baseline***: Global comm to find AMR block neighbors
2. ***In situ refined***: Gets AMR block neighbors from CTH.
3. ***In transit extra***: Extra nodes used for VDA service
4. ***In transit internal***: Carve out nodes for analysis (less cores for CTH)
5. ***Disk-based post processing***: Traditional approach

In transit extra



In transit internal

In transit and post-processing workflows use baseline algorithm.

Experimental Setup

- System: Cielo supercomputer at LANL
  - 8,944 node Cray XE6 (1.37 Petaflops peak)
  - Node: 2 AMD Opteron 6136 (Magny-Cours) 8-way processor chips
  - 32 GB memory/node
- Application: CTH (AMR) + Catalyst
  - 500 time steps of CTH
  - 51 analysis steps (approximately once every 10 time steps)
  - Five application workflows from previous slide
- Experiments
  - Strong scaling for three datasets: 33k blocks, 218k blocks, 1.5m blocks
  - Five runs for variance data
  - Data captured from instrumented code and HPCToolkit
  - Over 10m node hours for development, debugging, experiments.

Choosing the Number of Service Nodes

Memory Requirements for In Transit Service

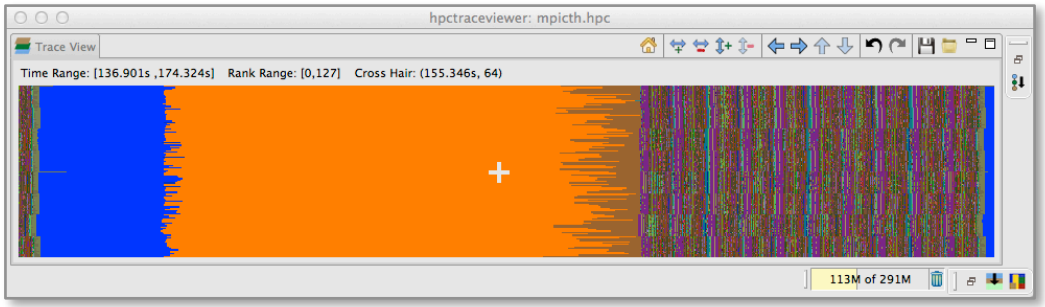
- Constraints given 32 GiB/node
  - Based on "trial and error" we found that one node can manage/process ~16K AMR blocks from CTH.

- Number of service nodes required for In Transit
  - 33k blocks: 2 nodes
  - 129k blocks: 16 nodes
  - 1.5m blocks: 100 nodes

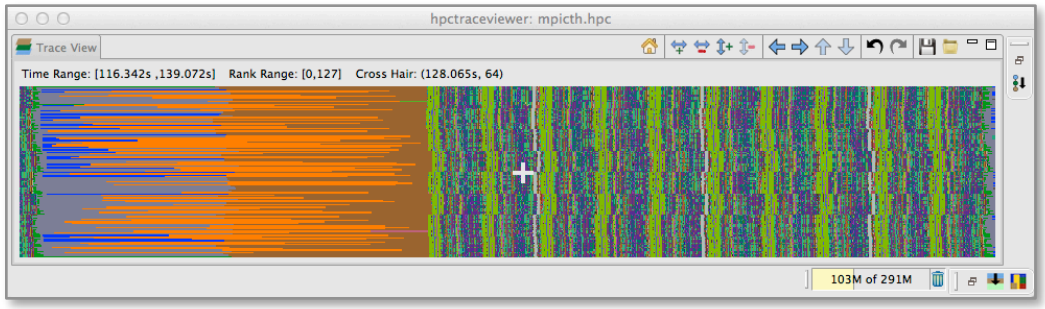
Choosing the Number of Service Nodes

Computing Requirements for In Transit Service

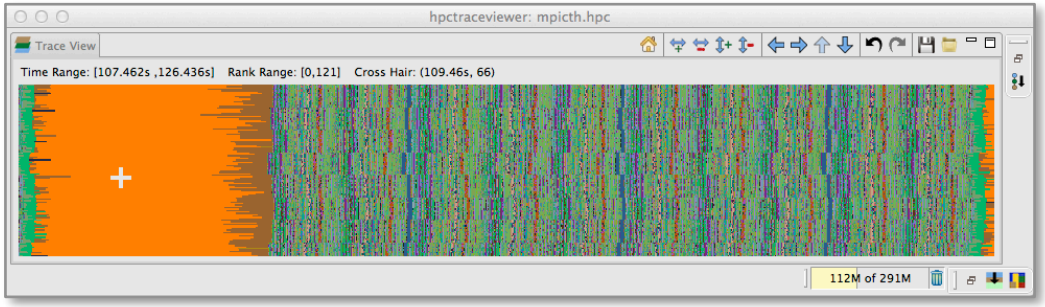
Wait for Server Transfer Data



2 server cores: 64:1
- 10 cycles in 37 secs
- Client idle waiting for servers (also affects xfers)



4 server cores: 32:1
- 10 cycles in 23 secs

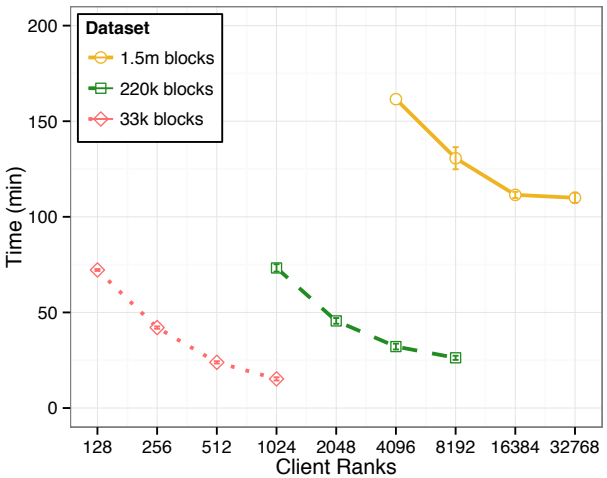


8 server cores: 16:1
- 10 cycles in 19 secs
- Less than 1% time waiting

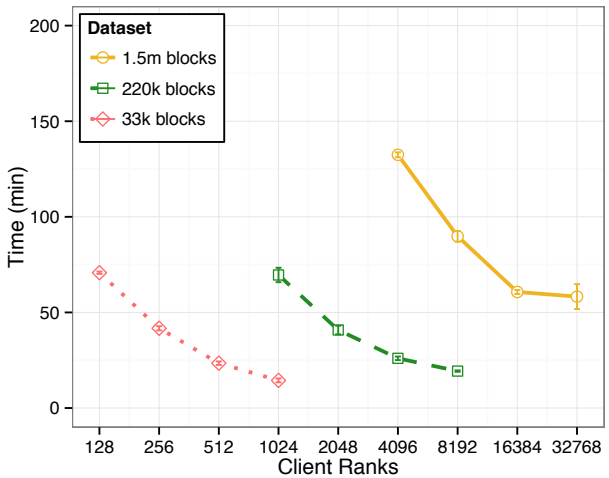
Allocations for Experiments

Data Set	CTH Client Cores (16/node)		In Transit Server Cores (8/node)
	In-situ (all), in-transit extra	In-transit internal	
33k blocks	128	96	16
	256	224	16
	512	480	16
	1,024	992	16
218k blocks	1,024	768	128
	2,048	1,792	128
	4,096	3,840	128
	8,192	7,936	128
1.5m blocks	4,096	2,496	800
	8,192	6,592	800
	16,384	14,784	800
	32,768	31,168	800

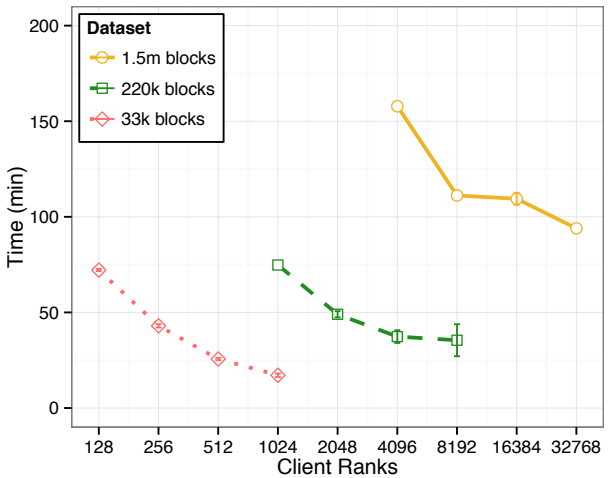
Total Runtime for All Experiments



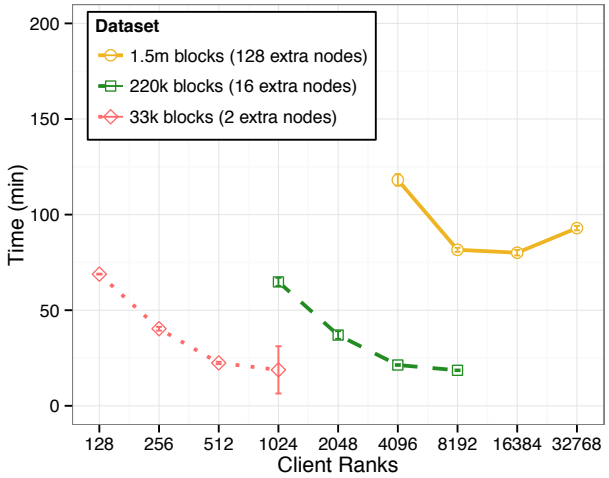
In situ baseline



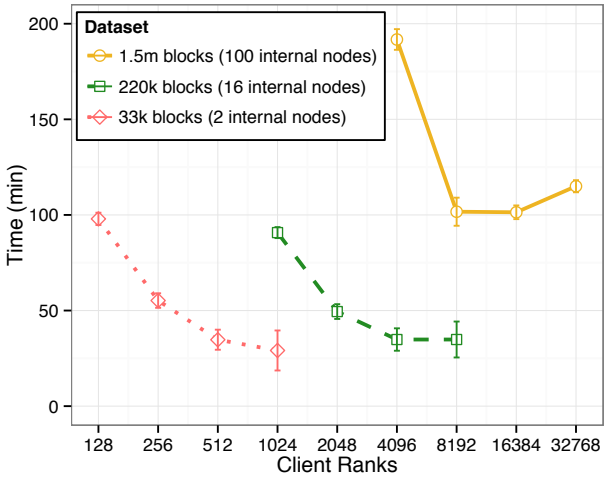
In situ refined



Disk-based post processing



In transit extra nodes



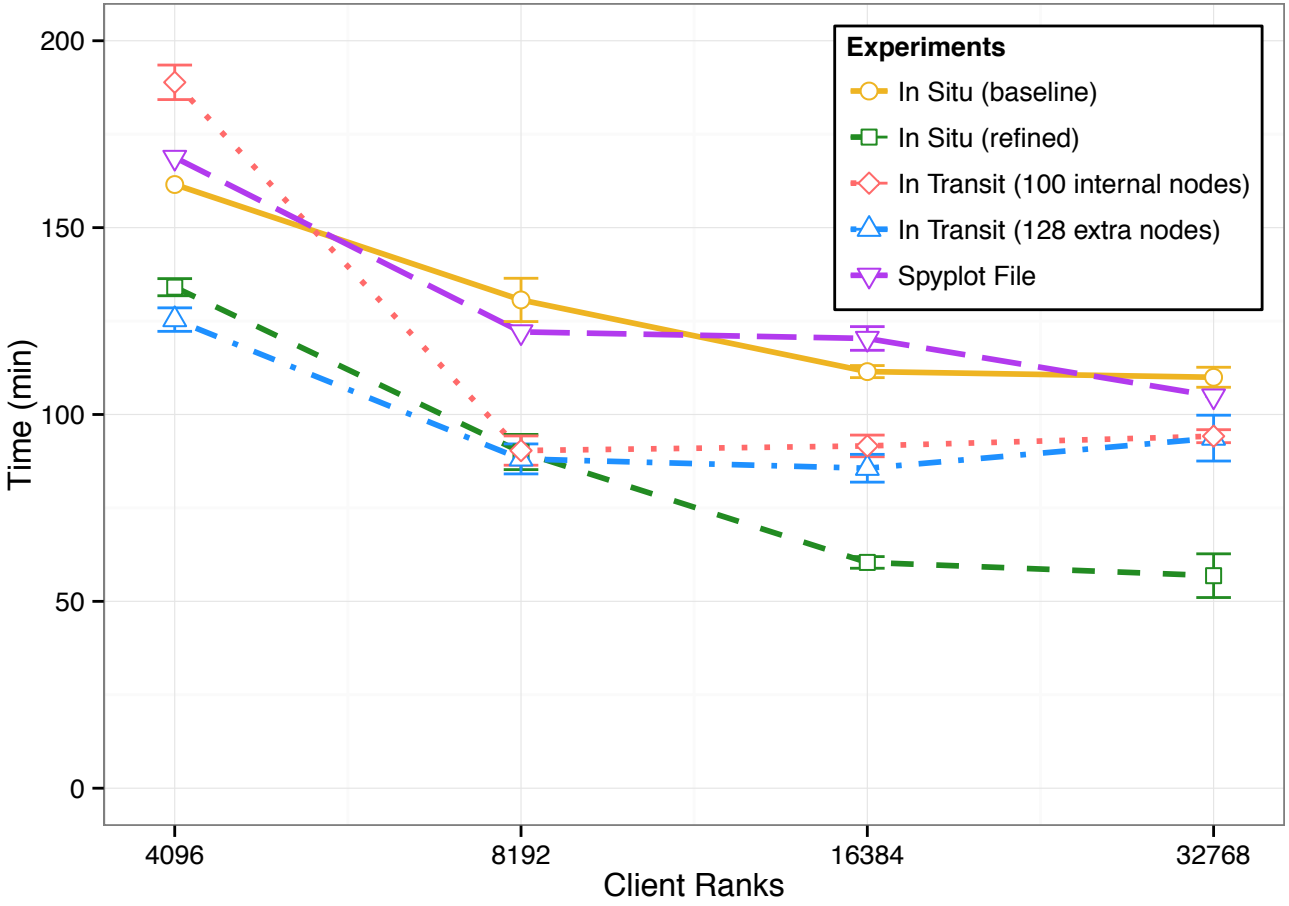
In transit internal nodes

5 applications
3 datasets

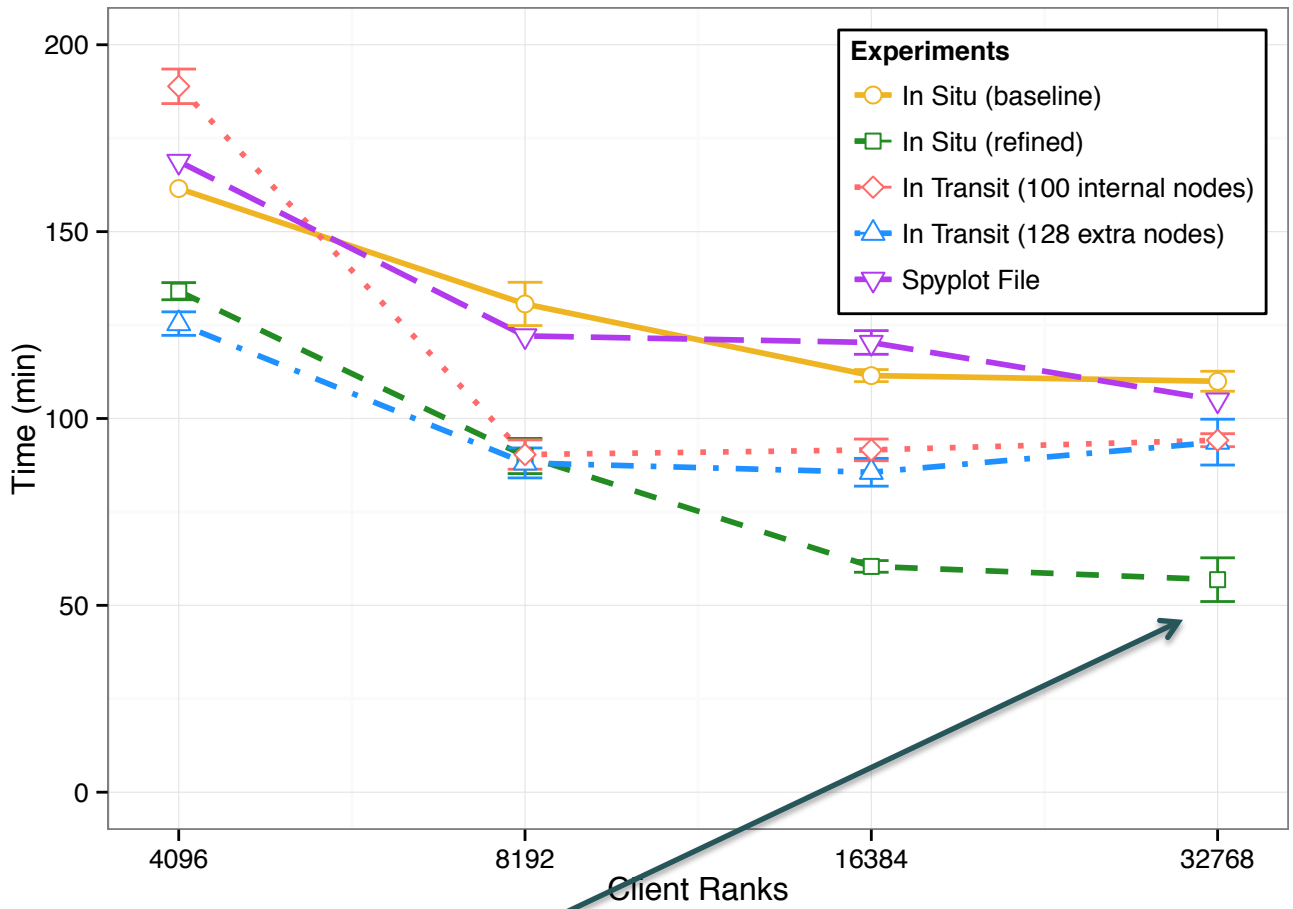
Strong scaling for each dataset

Error bars show variance

Summary Timing (1.5m blocks)

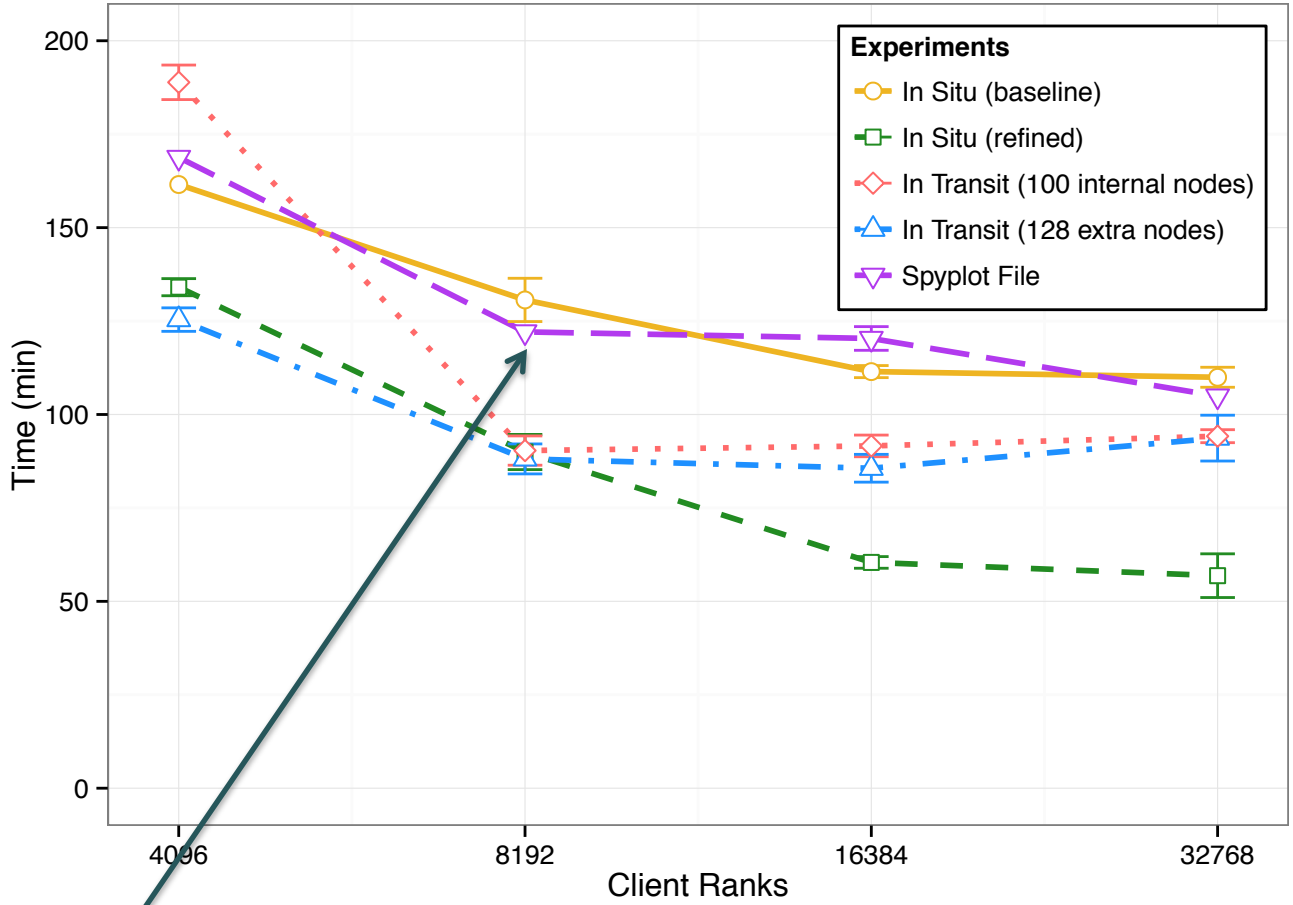


Summary Timing (1.5m blocks)



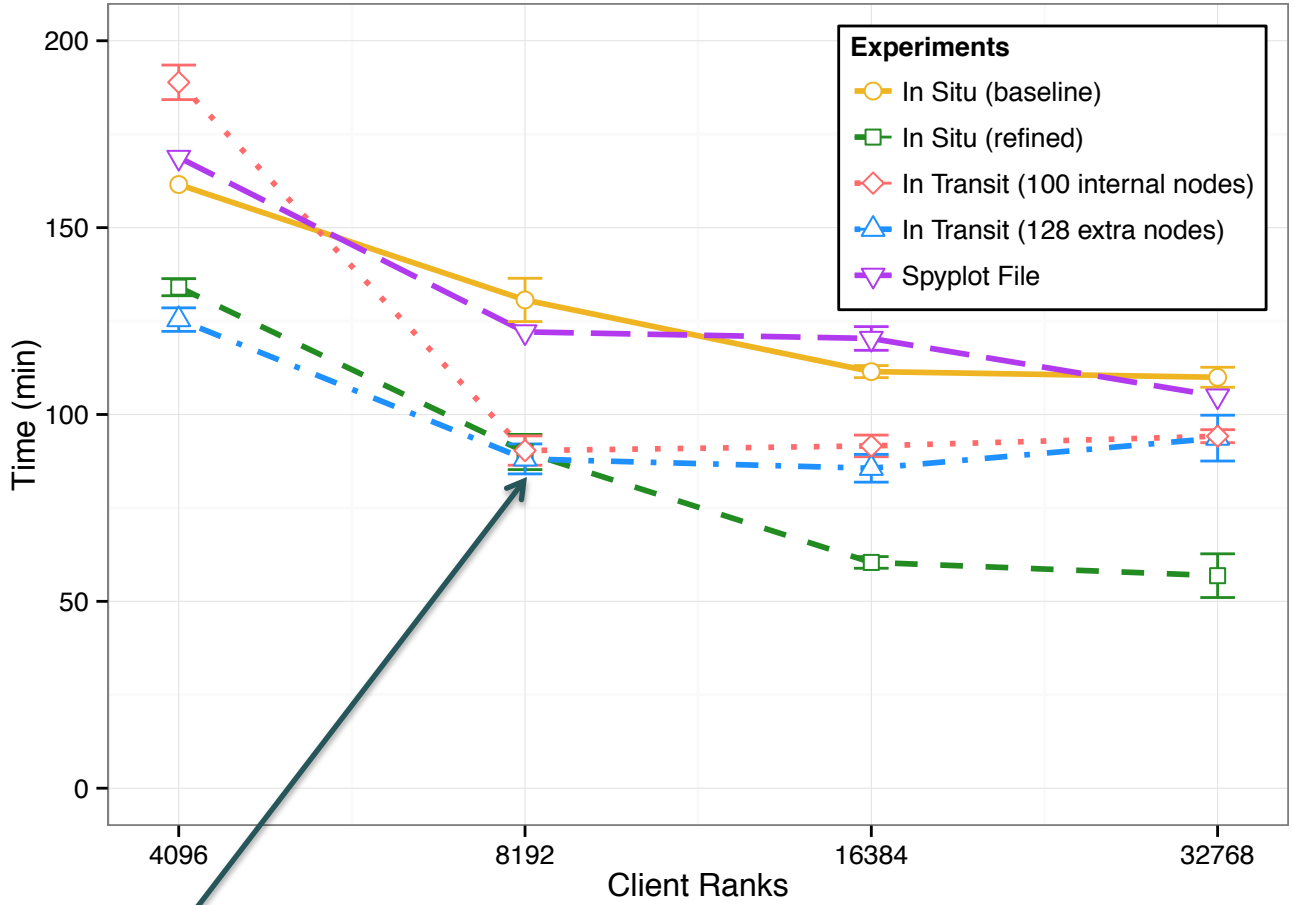
No significant improvement at 32K cores. Probably insufficient work for analysis (only 45 blocks per process).

Summary Timing (1.5m blocks)



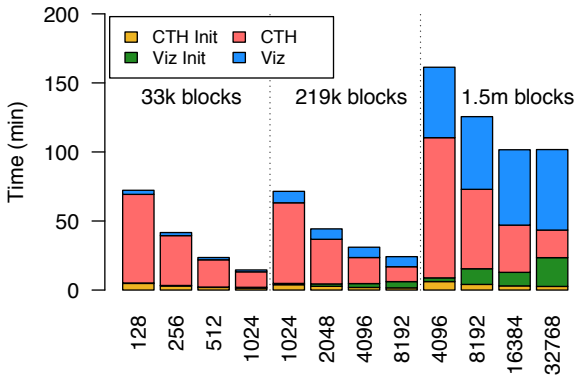
Writing files surprisingly fast. Although slower than most alternatives, still a viable option.

Summary Timing (1.5m blocks)

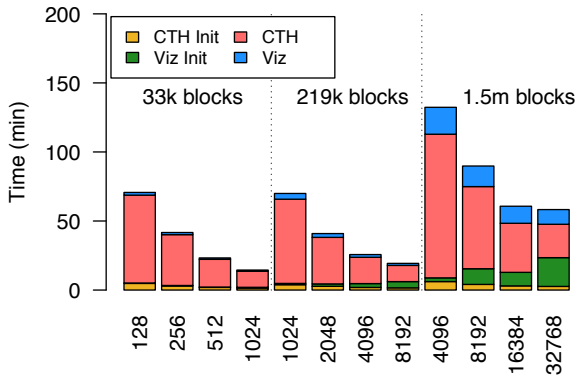


"Sweet spot" at 8K cores: *in transit* with unrefined algorithm equal to *in situ* with refined algorithm.

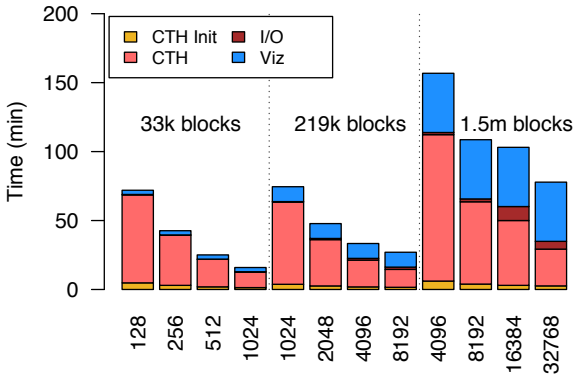
Timing Per Task



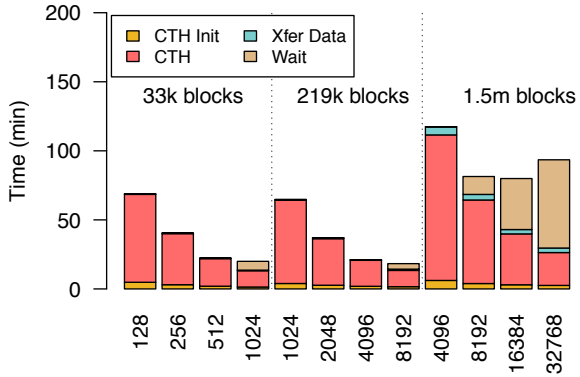
In situ baseline



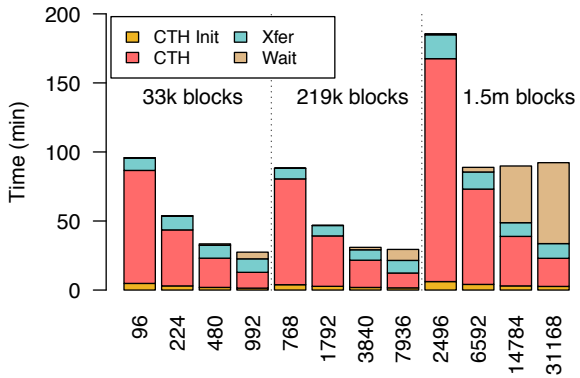
In situ refined



Disk-based post processing



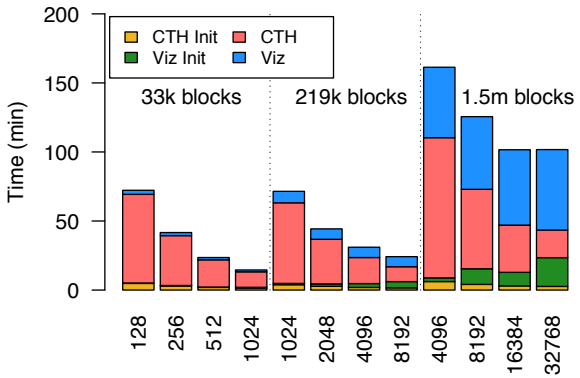
In transit extra nodes



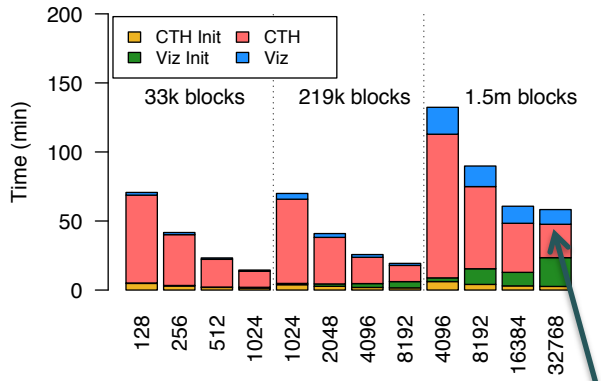
In transit internal nodes

- CTH scales well.
- Baseline algorithm does not scale
- Spyplot I/O not bad

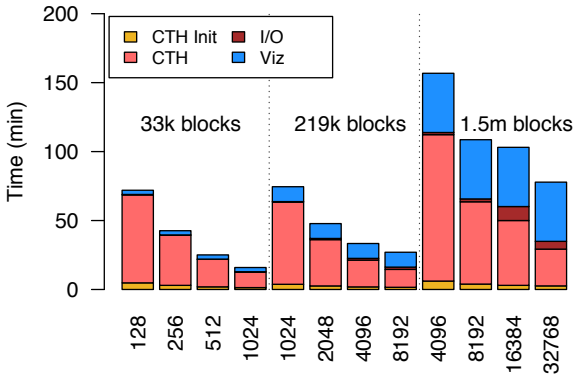
Timing Per Task



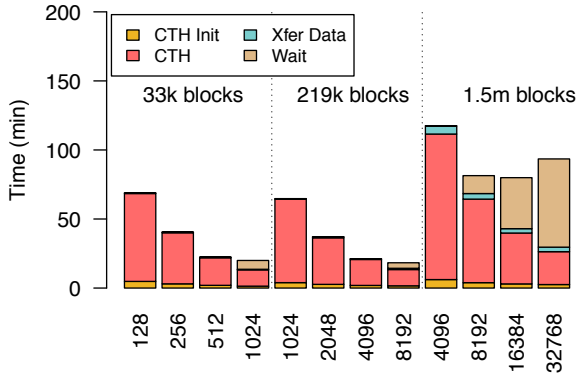
In situ baseline



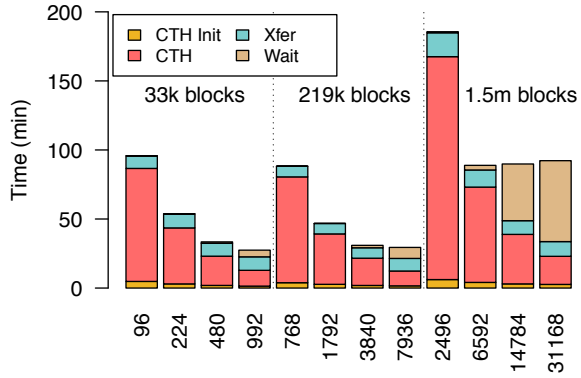
In situ refined



Disk-based post processing



In transit extra nodes

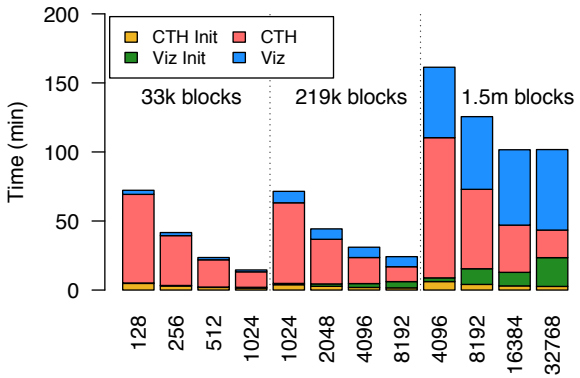


In transit internal nodes

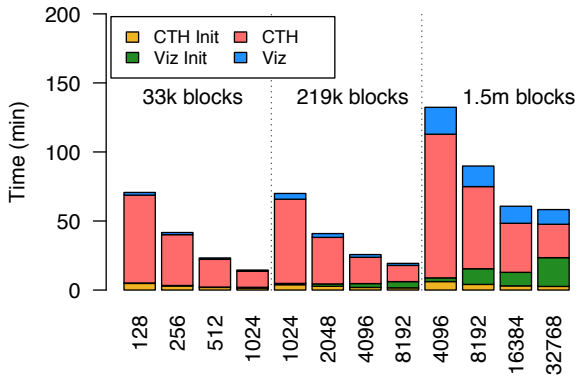
Refined analysis has lower overhead, but initialization is problematic.

Refined algorithm requires additional data to be passed. Not done for *in transit* experiments.

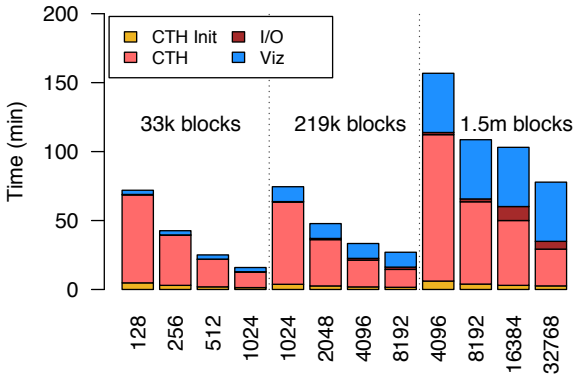
Timing Per Task



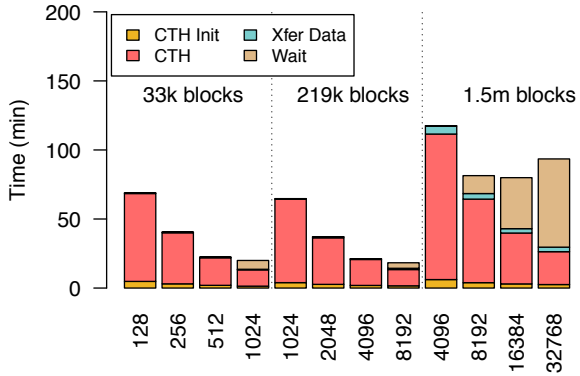
In situ baseline



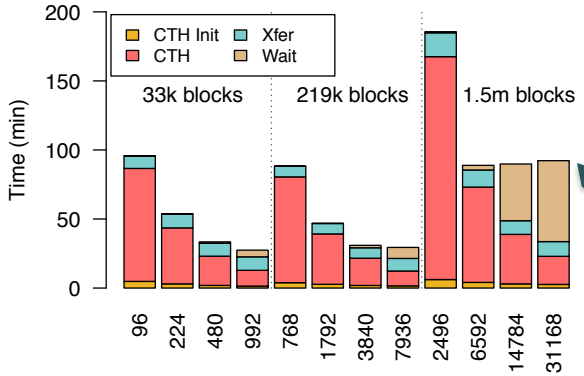
In situ refined



Disk-based post processing



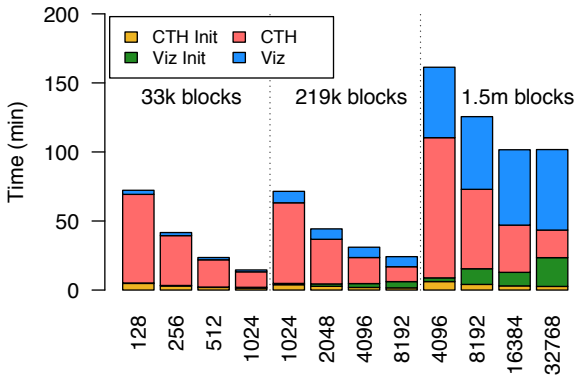
In transit extra nodes



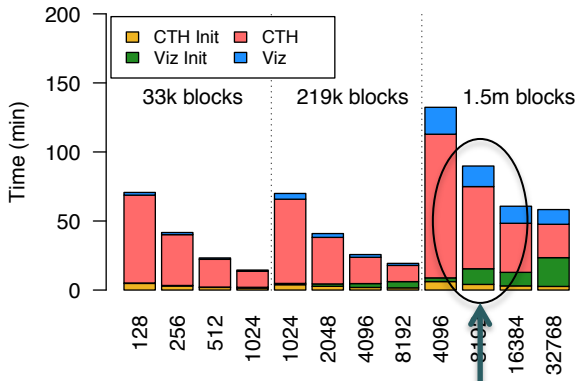
In transit internal nodes

Service is a fixed size (100 nodes), the wait time should be independent of the number of cores on the client.

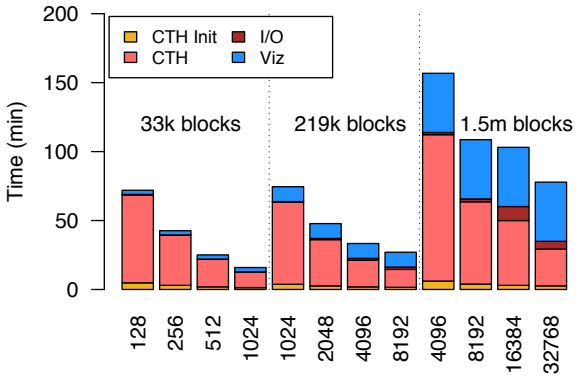
Timing Per Task



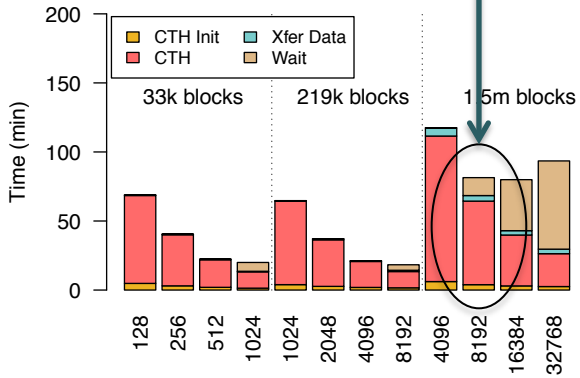
In situ baseline



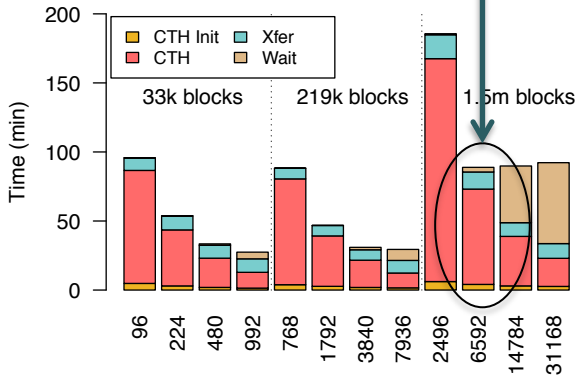
In situ refined



Disk-based post processing



In transit extra nodes



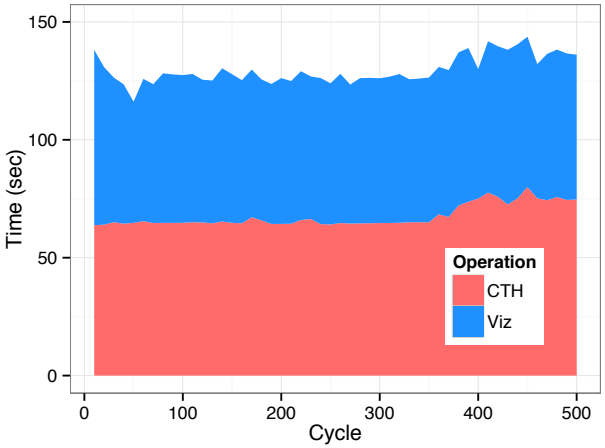
In transit internal nodes

"sweet spot"

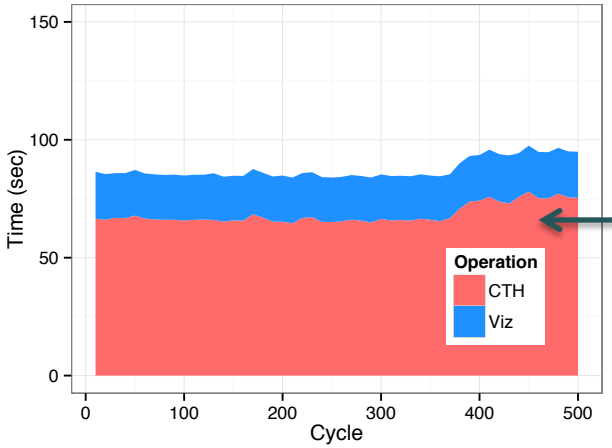
In transit internal shows balanced simulation and analysis

Time-Series Analysis (8k cores)

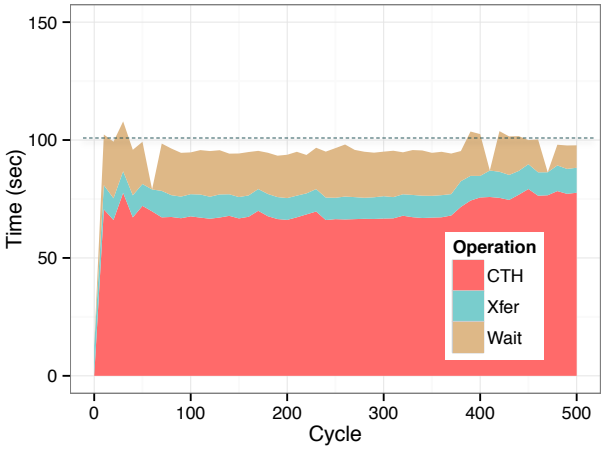
10-cycle increments



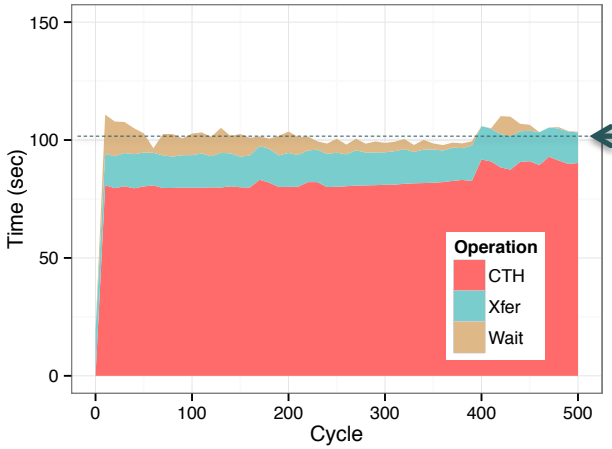
In situ baseline



In situ refined



In transit extra nodes



In transit internal nodes

Although number of blocks changes very little, CTH runtime gets longer as simulation progresses.

Vis time is roughly constant.

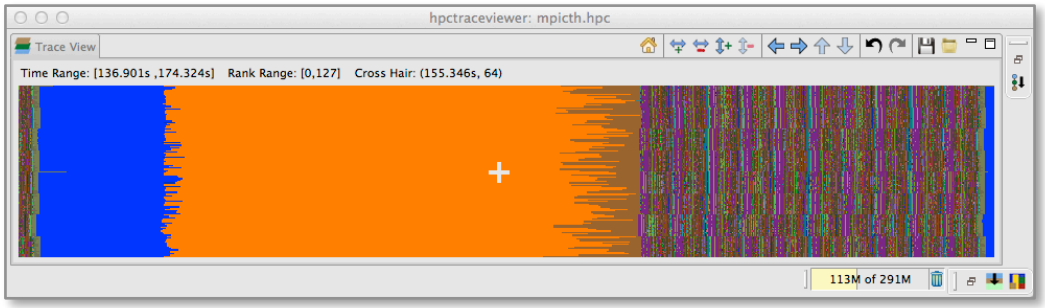
In transit "wins" when xfer+wait is less than viz.

In transit can flatten the runtime as long as extra simulation time consumes only wait time.

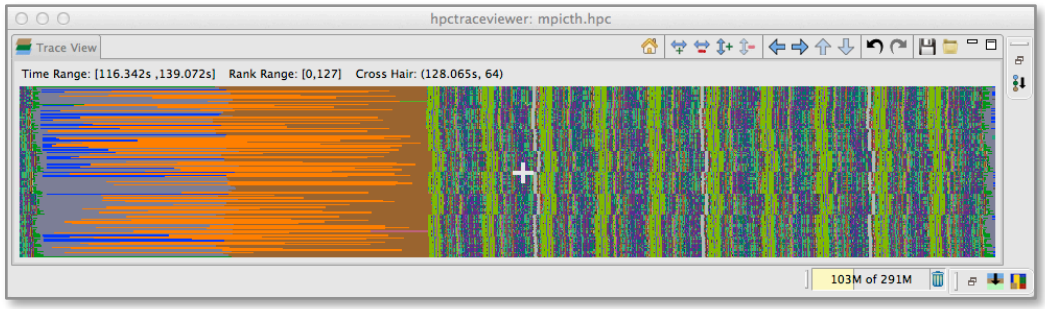
In Transit Node Scaling

Recap preliminary results

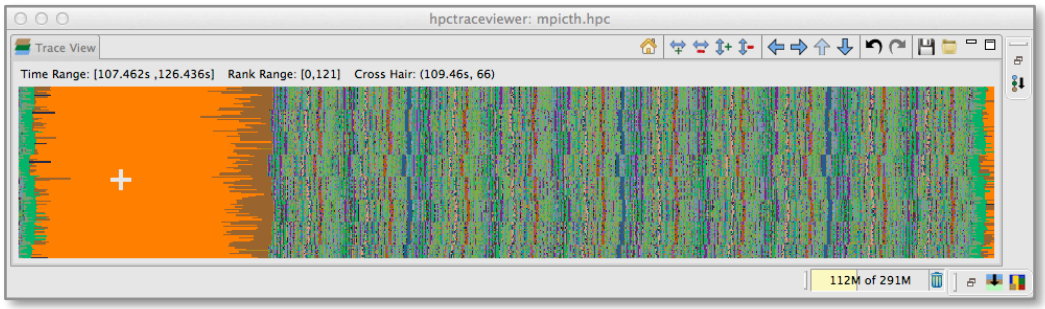
Wait for Server Transfer Data



2 server cores: 64:1
- 10 cycles in 37 secs
- Client idle waiting for servers (also affects xfers)

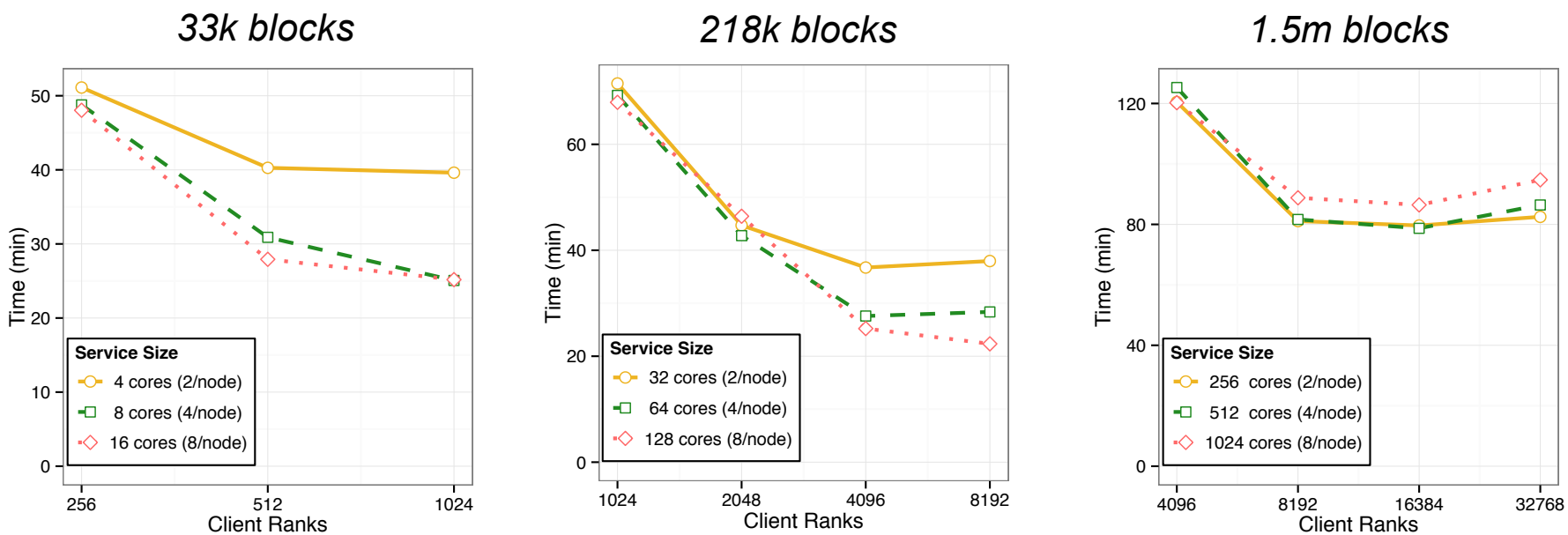


4 server cores: 32:1
- 10 cycles in 23 secs



8 server cores: 16:1
- 10 cycles in 19 secs
- Less than 1% time waiting

In Transit Service-Node Core Scaling



For small datasets, there is clear benefit to using 4 and 8 cores/node (agreement from preliminary tests)

For the 1.5m blocks dataset (at large scale), the opposite appears to be true. Needs further study.

Conclusions

- In situ is extremely effective when analysis algorithm scales with the simulation code.
- In transit is beneficial for complex cases, where data-transfer (and wait) costs less than analysis.
- Balance is the key. Efficient use of resources requires careful consideration of memory, compute, and network requirements of both simulation and analysis codes.
- Traditional disk-based post-processing approaches are not dead… yet.
- Better system support is needed for in-transit approaches. Scheduling is a challenge and node sharing is not possible.

Future Work

- Evaluate algorithms with different scaling behaviors
  - Contour algorithm (perfectly scalable)
  - Refined water tight contours (reasonably scalable)
  - Baseline water tight contours (not scalable)
- Integrate application knowledge for in transit
  - Knowledge of memory layout on client allows zero-copy transfers
  - Knowledge of mesh structure allows scalable (refined) alg on service.
- No-wait analysis (*in transit*)
  - Perform analysis if and only if the service is ready
- Additional apps at Cielo scale
- Improved OS and runtime support
  - Scheduling, placement, node sharing, specialized runtimes, …

Integrating Analysis and Computation:

Experiences with In-Situ and In-Transit Approaches

Approved for Public Release: SAND2013-XXXXP

Ron A. Oldfield, David Rogers,
Ken Moreland, and Nathan Fabian

Sandia National Laboratories
Albuquerque, NM, USA

SOS 17
Jeckyll Island, GA
March 2013

Sandia National Laboratories

Exceptional service in the national interest

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.