Title: Coupled Application+System Performance Profiling

Author(s): Wofford, John Quince III

Intended for: Would like to make slides available for students to download and use at home organization.
Web

Issued: 2019-08-15

Coupled Application+System Performance Profiling

New Darwin tools

Quincy Wofford

07/14/2019

module load ovis/4.2.3

What is it?

Lightweight Distributed Metric Service (LDMS) v4.2.3

What does it do?

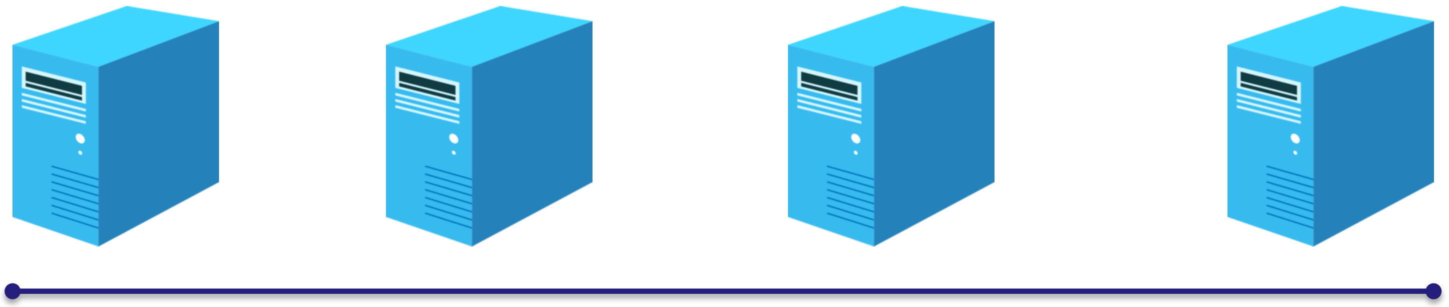
Samples /proc, /sys, /dev/shm mounts on the order of 1 second. Guaranteed “happened before” relationship between time indicies *across all sampler nodes*. Unprivileged user ok.

Why is this new?

Trinitite, Fog have LDMS v3, which doesn’t support application performance profiling.

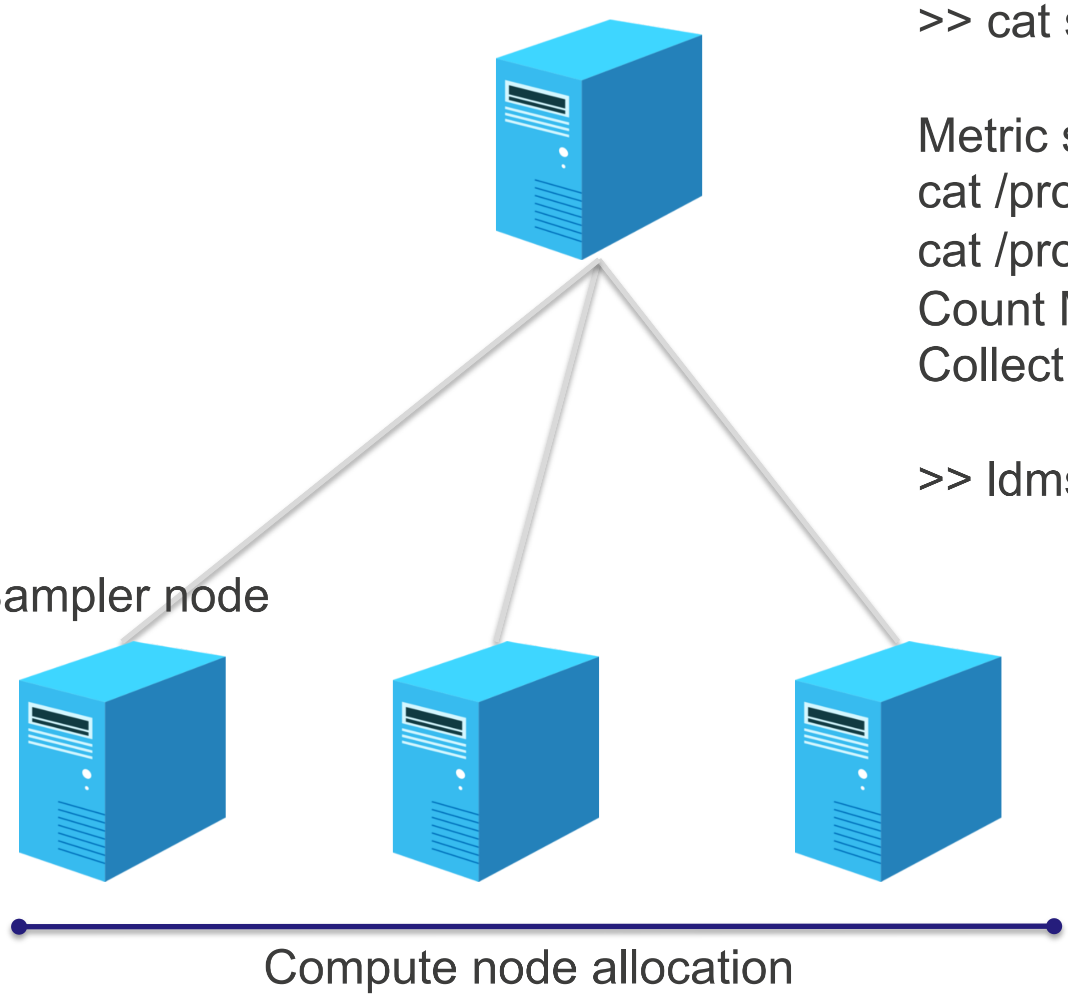
Lightweight Distributed Metric Service (LDMS) v4.2.3

Obtain allocation



Compute node allocation

Lightweight Distributed Metric Service (LDMS) v4.2.3

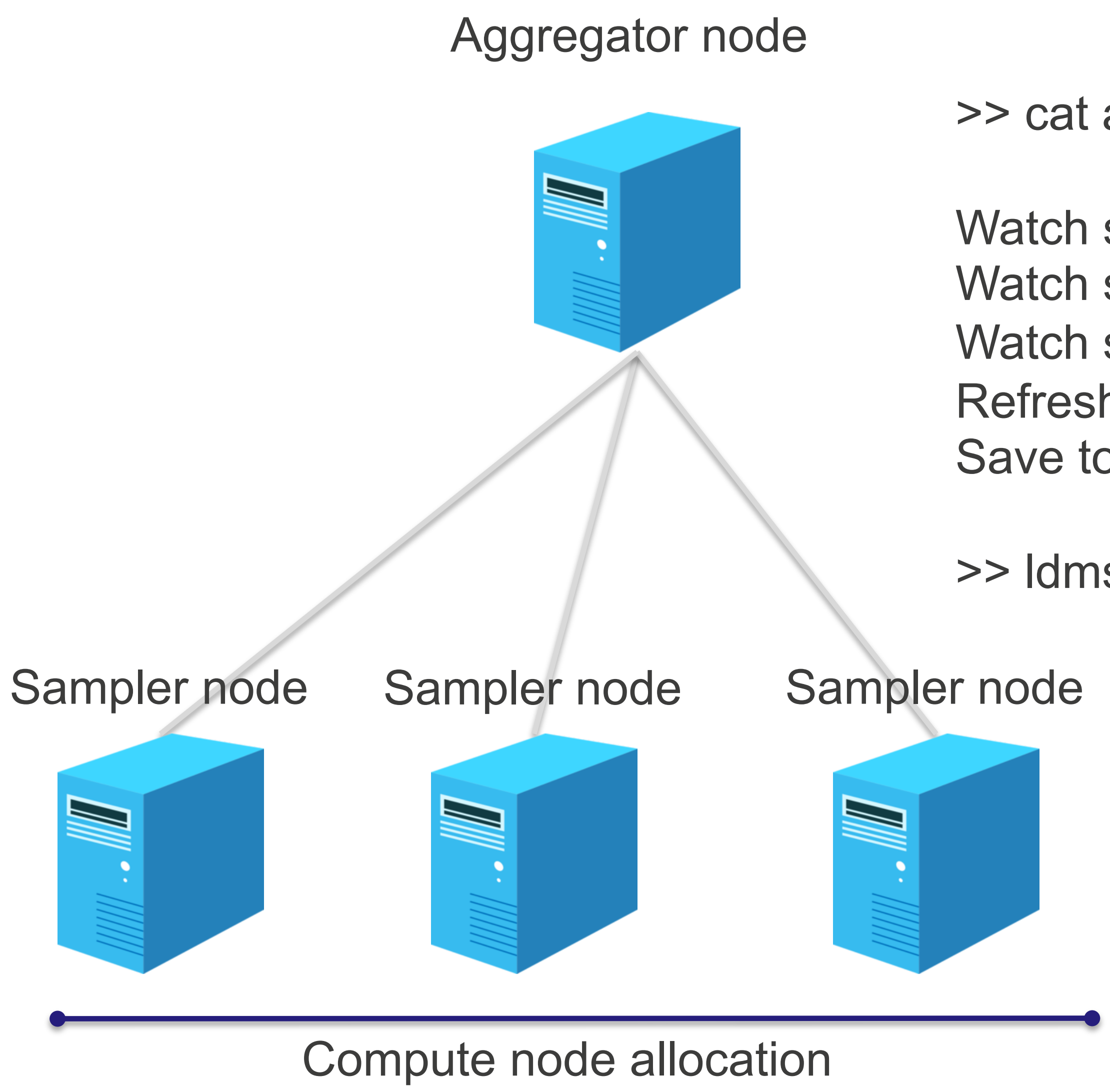


```
>> cat sampler.conf

Metric set = {
cat /proc/meminfo,
cat /proc/stat,
Count MPI_Send }
Collect every 1 sec

>> ldmsd -c sampler.conf
```

Lightweight Distributed Metric Service (LDMS) v4.2.3

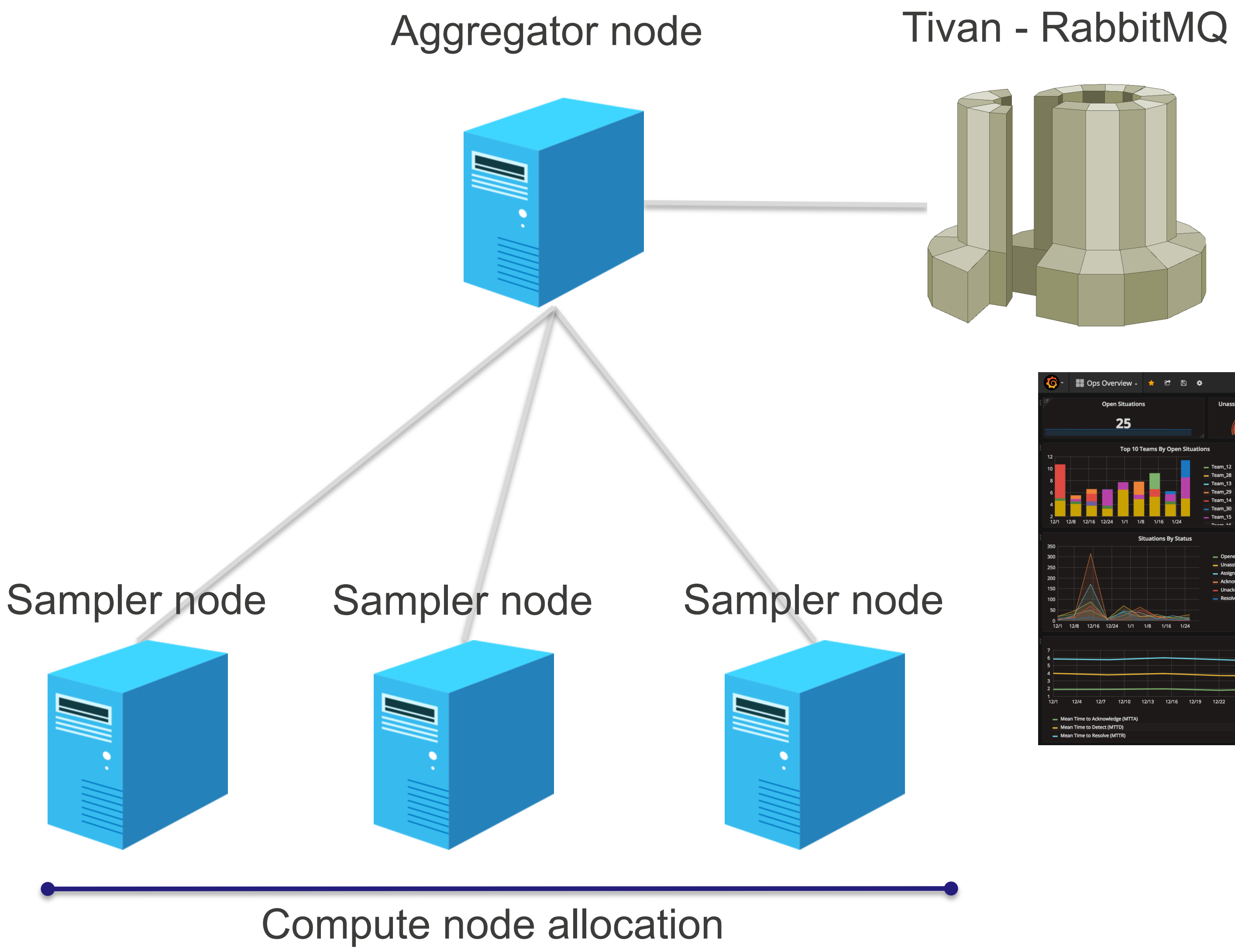


```
>> cat aggregator.conf

Watch sampler1
Watch sampler2
Watch sampler3
Refresh every 10 sec
Save to csv, or send to Tivan

>> ldmsd –c aggregator.conf
```

Lightweight Distributed Metric Service (LDMS) v4.2.3



Lightweight Distributed Metric Service (LDMS) v4.2.3

What does this buy me?

OpenMPI as a proxy application for your distributed parallel jobs.

- Detect patterns in workload app => anomaly detection
  “Production App. Perf. Data Streaming for Sys. Monitoring” Izadpanah R., Allan BA., et al.
- No need to instrument all your codes if communication behavior is primary interest, but you can if needed.

Synchronous system+application stats

- There was a power failure last night. My code crashed last night. Did my code crash at the same time as the power failure, or before?
- How hot were the NICs on each of my nodes when I started observing poor collective performance?

Lightweight Distributed Metric Service (LDMS) v4.2.3

How can I get started?

Docker: https://hub.docker.com/r/qwofford/ldms

su qwofford; cd; ls;

Contained within is: sampler.conf, start_sampler.sh, and run_mpi_test.sh

Darwin:

module load ovis/4.2.3;

module show ovis;

Explore “configs” directory in the project root. (First line after module show)
Contained within is : agg_rabbit.conf, agg_csv.conf, sampler.conf, start_sampler.sh, start_agg.sh, and a README.

Application+System Performance Profiling for HPC

Coupled application+system profiling at scale

Quincy Wofford

07/14/2019

~~What is the problem?~~

- “I want to run this application faster.”
- “I want to fit this application into memory.”

~~What is the problem?~~

- “I want to run this application faster.”
- “I want to fit this application into memory.”

Not problem statements!

(…but they certainly indicate a problem)

How can *I* find the problem statement?

Questions we *might* should ask (new/small projects):

- What is the value of this application?
- What is an acceptable runtime?
- Is there something simpler that runs faster?
- What does a “correct” ending state look like?

Questions we should *always* ask:

- Where is it running?
- How fast does it run now?
- If distributed-parallel, what is a useful scale?
- If scientific simulation, what is a representative input deck?

How can *I* find the problem statement?

Questions we *might* should ask (new/small projects):

- What is the value of this application?
- What is an acceptable runtime?
- Is there something simpler that runs faster?
- What does a "correct" ending state look like?

Questions we should *always* ask:

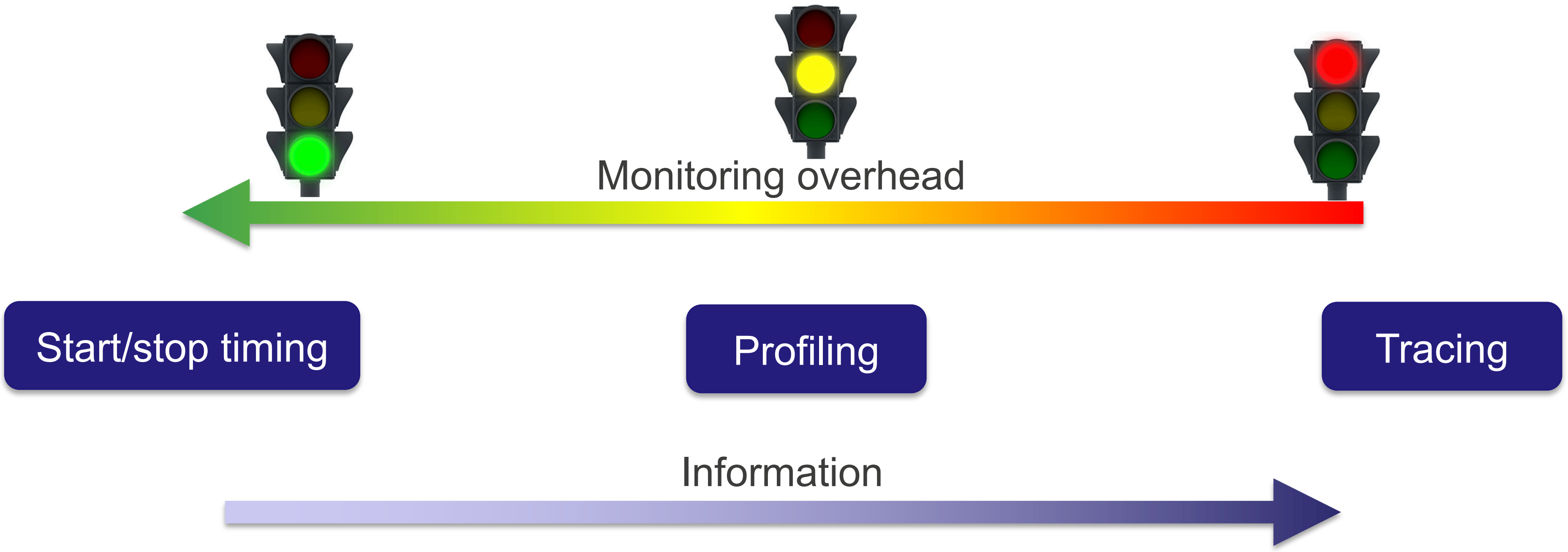
- Where is it running? Darwin, scaling partition
- How fast does it run now? 2 hours
- Distributed-parallel, what is a useful scale? 1000x1000x1000
- Scientific simulation, what is a representative input deck?
  MiniMD < in.lj.miniMD

How can *I* find the problem statement?

Performance assessment.

Production performance := Application + System(s) overhead

Monitored performance := Application + System(s) overhead + Monitoring overhead



How can *I* find the problem statement?

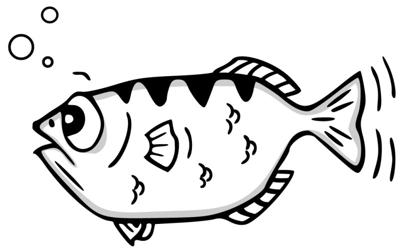
Performance assessment.

Step 1

Which function calls are taking the most time? -> Profiling

Step 2

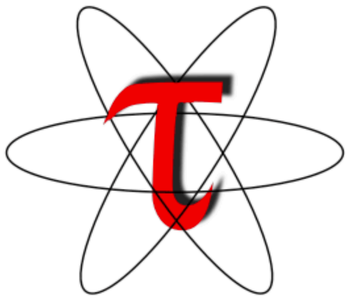
Why are the the most expensive N functions taking so long? -> Tracing



GDB

Tools

TAU



How can *I* find the problem statement?

Performance assessment workflow.

1. Run small problem size, serially (~seconds). Profile or time.
2. Run medium problem size, serially (~minutes). Profile or time.
3. Validate outputs. Observe scaling behavior.
   "Does it even work?"
4. Run medium problem size, "modestly" parallel (~minutes). Profile.
5. Run medium problem size, increase parallelism (~minutes). Profile.
   "Strong scaling, is the algorithm any good?"
   1. Yes. Go to 6.
   2. No. Trace to learn why.
6. Run small, medium, production problem size with constant problem size:parallelism ratio.
   "Weak scaling, how does the algorithm perform at scale?"
   1. OK. Stop.
   2. Poorly. Trace to learn why.
7. Deliver.