Pilot, the Benchmarking Component of the ASCAR Automatic Storage Tuning Framework

Yan Li
Dec. 2, 2015
Center for Research in Storage Systems (CRSS)
University of California, Santa Cruz
Overview of the ASCAR Project

**ASCAR:** ASCAR Storage Control and Regulation System (starting from 2013)

**Goal:** Tuning and optimizing computer systems automatically without human supervision

**Value proposition:**
- Better performance than human tuning
- Faster deployment and easier monitoring
- Reduce human cost

**How:** machine learning and heuristics
What we have done so far:
A prototype that works with Lustre

Highlights:
Increases performance for all workloads we’ve tested by 2% to 35% (Li, MSST’15)

We have released the code to promote collaboration:
https://github.com/mlogic/ascar-lustre-2.4-client
https://github.com/mlogic/ascar-lustre-sharp
What kept us busy in the past six months

Producing trustable benchmark results
Trustable benchmark is crucial to systems research and engineering

Developers rely on benchmark to optimize system design and code

Consumers rely on benchmark to choose products

Technical support needs benchmark to diagnose and tune systems

It is that simple:
If your benchmarks are wrong, your conclusion is not likely to be correct.
Requirement of trustable benchmark results and how ASCAR addresses them
**Precision**: is your measurement a good approximation of the real performance?
1. Warm-up phase detection and removal using first derivative or moving average

![Graph showing Seq. Write Throughput]

Write throughput (MB/s)

Write request

sustainable performance we want to measure
2. Confidence interval (CI) analysis

What you want: true mean ($\bar{u}$)
What you have: sample mean ($\bar{u}_n$)

We need to make sure that $\bar{u}_n$ is not too far from $\bar{u}$:

$$\text{Prob}(\left| \bar{u}_n - \bar{u} \right| < e) \geq K$$

$K$ is called the confidence level.

$e$ is called the confidence interval of $\bar{u}_n$. 
In LADS, we observe the maximum throughput at around 400-450MB/s for the experiment of a big data set, which is reasonable based on our test-bed configuration. The block-level throughput for all 16 disks is 2.3GB/s, the file system overhead reduces that by about 40% to 1.3-1.4GB/s. We tested with up to eight threads reducing the optimum to 650-700MB/s. Given thread synchronization overhead, 400-500MB/s is reasonable but improvement is still possible.
3. Determining the required sample size (or benchmark duration)

is this long enough to get a small enough confidence interval at 95% confidence?
3. Determining the required sample size (or benchmark duration)

You have to run your benchmark long enough (or as many times as necessary) to produce a sample mean \( \bar{u}_n \) and a CI for a sufficiently high \( K \) (usually 95%).

For normally distributed data, you can use Student’s \( t \) distribution.
However,

Student’s $t$ dist. requires normally distributed data

Most measurements in computer systems are not normal. Some follow a long tail log-normal distribution.
4. Normalization
(or how to calculate CI for non-normal samples)

Arithmetic mean should be normal if the sample is sufficiently large according to **Central Limit Theorem (CLT)**

And 20, 50, 100 are not large enough. [Char HPCA’12] suggests 500 at least.

Test for normalization wherever you can
Q-Q graph, Royston-Shapiro-Wilk test
However,

CLT requires i.i.d (identically and independently distributed) sample

Actually, most systems’ benchmark results are highly autocorrelated.
Independent and identically distributed is very important

Both Central Limit Theorem (CLT) and $t$-distribution require that samples are i.i.d.

Must check autocorrelation coefficient in your test data

\[ R_1 = \text{Cov}(u_i, u_{i+1}) = E[(u_i - \bar{u}_n)(u_{i+1} - \bar{u}_n)] \]
Sample autocorrelation coefficient calculation

Workload:
Sequential write to a shared network drive
I/O size: 4 KB
Total length: 300 MB

autocorrelation coefficient: 0.614
(0 is no autocorrelation, 1 is full autocorrelated)
4. Normalization
(or how to calculate CI for non-normal samples)

We can merge adjacent samples (to subsessions) to reduce autocorrelation.

Keep merging until the autocorrelation coefficient is reduced to less than a threshold.
A subsession can be very large

Workload:
Sequential write to a shared network drive
I/O size: 4 KB
Total length: 300 MB

autocorrelation coefficient: 0.614

optimal subsession size (q): 10193
5. Outlier detection (autocorrelation analysis also helps to reject borked rounds)
There’s more in addition to precision

**Accuracy:** the result reading should measure what you think the workload measures, i.e., the performance reading is not limited by some other bottleneck.

**Overhead analysis:** the overhead of the measurement mechanism must be clearly documented

**Comparability:** confidence interval and variance must be taken in consideration when comparing benchmark results.
The pilot tool

https://pilot-tool.slack.com
https://github.com/mlogic/pilot-bench

Current features:
1. warm-up phase removal
2. optimal benchmark duration calculation
3. i.i.d. test and using independent replications
4. result export and retrieval
5. result plotting
6. C/C++ binding

Planned features:
1. various normalization test
2. easier to use interface
3. Python binding
Live Demo
Plan for the following six month

- Implementing the planned features of pilot
- Porting the ASCAR Lustre prototype to the latest pilot release
- Building a prototype for automatic performance benchmark of Ceph using pilot
- Starting the implementation of ASCAR algorithms on the latest Lustre and Ceph (if time permits)
The team

**Faculty & Staff**
Darrell Long
Ethan Miller
Andy Hospodor

**Students**
Yan Li
Kamala Ramasubramanian
Rahul Talari
William Kong