Performance Measurement and monitoring in TSUBAME2.5 towards next generation supercomputers

aXXLs workshop @ ISC-HPC 2015, Jul 16, 2015

Akihiro Nomura

Global Scientific Information and Computing Center

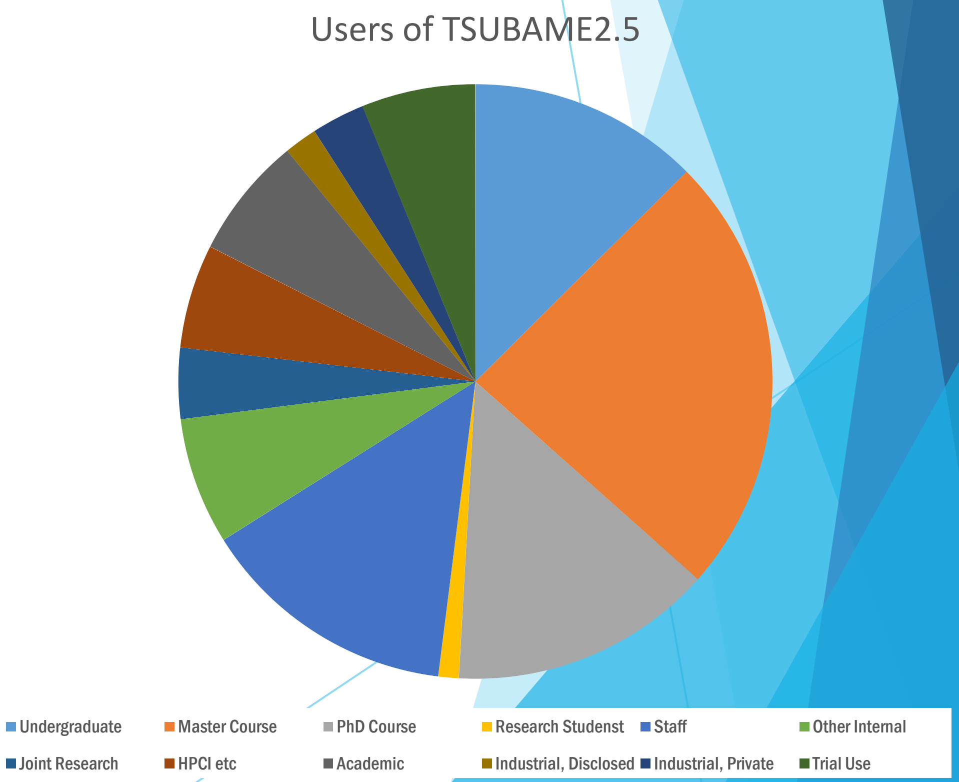
Tokyo Institute of Technology

Why we collect performance data?

- As Application developer / user
  - Improve performance of my application
    - Know what component is the ***bottleneck***
    - See whether we did ***something stupid***
    - …
- As Supercomputing service provider ← *I'll talk as a provider today*
  - Help improving performance of user applications → act as ***Evangelist***
  - Unveil what is actually going on in supercomputer
    - ***Optimize*** computer's ***settings*** to serve resource efficiently
    - Use statistics to ***design*** next-generation supercomputer

TSUBAME2.5 in Tokyo Tech

- GPU-based cluster with 1408 nodes (+ Fat memory nodes)
  - CPU: Intel Xeon X5670 (westmere) x 2
  - GPU: NVIDIA TESLA K20X x 3
  - Memory: 56GB ~ 96GB
  - Interconnect: InfiniBand QDR x 2 (Injection BW: 80Gbps)
    - Connected to full-bisection dual-rail fat-tree network
  - SSD: 120GB ~ 240GB
  - Storage: Lustre and GPFS w/ HSM
  - OS: SLES11 SP3
  - Scheduler: PBS Professional
- Active users: ~750
  - 1/3 of users are external users, including industrial users



Application-level measurement tools tested/available in our TSUBAME2.5

- Profiler / Tracer
  - Score-P (Scalasca, Vampir), Tau
    - Time, Visit, MPI Comm, GPU events, Performance counters from PAPI...
  - Exana
    - Memory access trace
- Library
  - PAPI
    - CPU performance counter, combines CUPTI and RAPL results
  - CUPTI
    - GPU performance counter
  - RAPL
    - Power consumption

Verifying the tools is important
Case study: PAPI counters in westmere

- PAPI offers some predefined metric
  - Calculated from RAW performance counter in each CPU
  - PAPI_L1_TCM, PAPI_SP_OPS, PAPI_DP_OPS, ...
- “Okay, let’s count FP operations to verify our performance model”
  - In theory, PAPI_SP_OPS + PAPI_DP_OPS gives the value
  - In reality, the sum of them were too large
- “So, let’s count total amount of memory accesses”
  - PAPI_L3_TCM (cache misses in LLC) × 64 (cache line size) ?
  - The counter value was 10 times fewer than what we expect

What's going on in those counters? FP counters case

- FP ops are calculated from # of FP ins and # of SSE(vector) FP ins
  - PAPI_SP_OPS = SSE_SINGLE_PRECISION + 3 × **SSE_FP_PACKED**
  - PAPI_DP_OPS = SSE_DOUBLE_PRECISION + **SSE_FP_PACKED**
- However, # of SSE ins does not distinguish precisions
  - SSE FP operations are double-counted
- Workaround 1: use appropriate precision and ignore others
  - Mixed precision?
- Workaround 2: prorate SSE ins contribution into SP and DP
  - SP_OPS = S + 3 × (S/S+D) SSE etc.
- Thank PAPI developers for identifying this problem

What's going on in those counters? Memory counters case

- Main memory access is not caused only by LLC misses
  - Prefetch
- No PAPI predefined counter for prefetch
- Workaround: Use HW dependent counters
  - Read: OFFCORE_RESPONSE_0:ANY_DATA_RD:OTHER:ANY_LLC_MISS
  - Write: OFFCORE_RESPONSE_0:ANY_DATA:OTHER:ANY_RESPONSE – Read
  - They empirically gives appropriate value for our test…
- Thank Intel researchers for identifying this problem

Documentation is required!

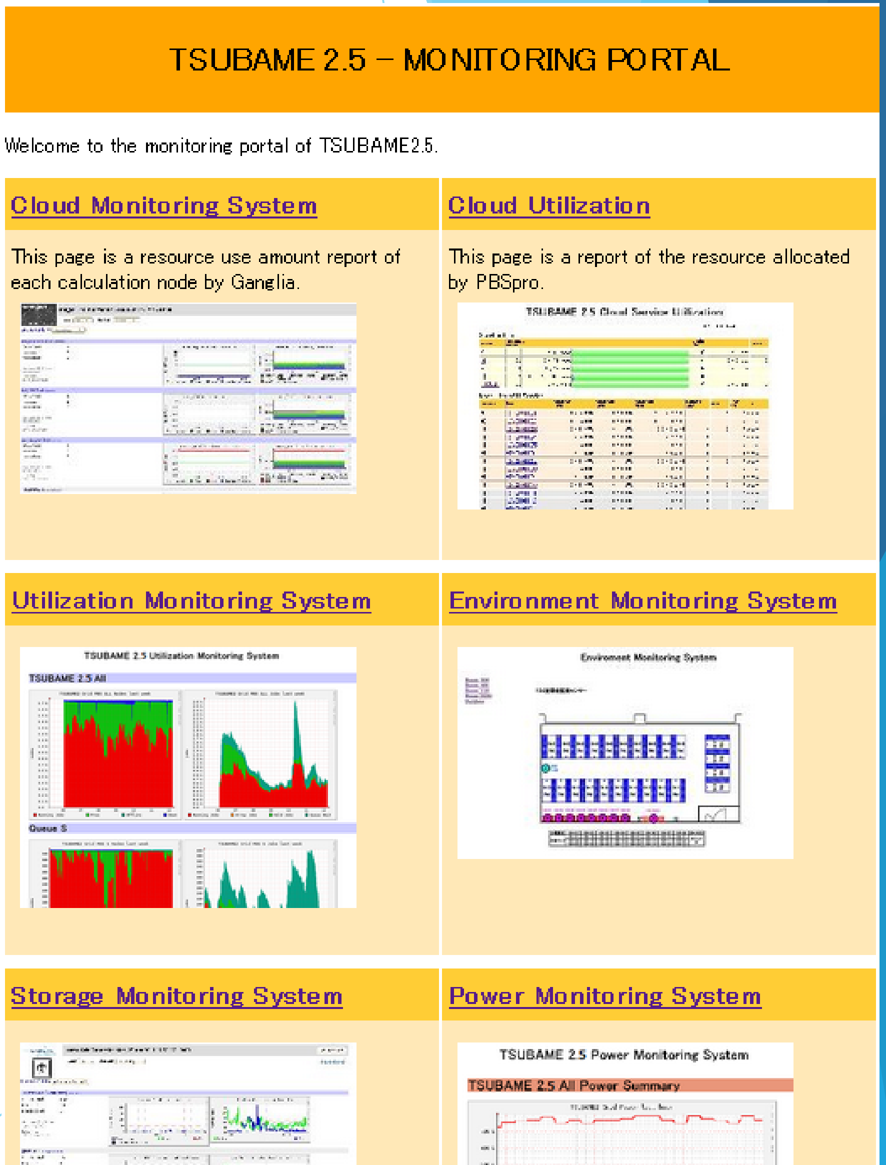
- In order to make the tools used by supercomputer users, we must provide documentation with
  - User’s native language (Japanese in this case)
  - Walkthrough with simple example, with verification
  - Workarounds with possible problems
- We provide some documents in TSUBAME website (experimental services)
  - http://tsubame.gsic.titech.ac.jp/en/labs-en
  - Walkthrough with Score-P, Vampir, Scalasca in Japanese
    - http://tsubame.gsic.titech.ac.jp/node/1245
- ~1 year after we start providing the documents, the tools started to be used by users (not by us or our collaborator)

System-level monitoring environment

- We are monitoring and logging the cluster's status
  - Node status (load, network, temperature, …) with ganglia
  - Power consumption at multiple levels
  - Storage status
    - 1 minute frequency, all the time of T2.5 operation (4.5 years)
  - Queue occupancy
  - Failure history (Node, NW, Power supply, …)

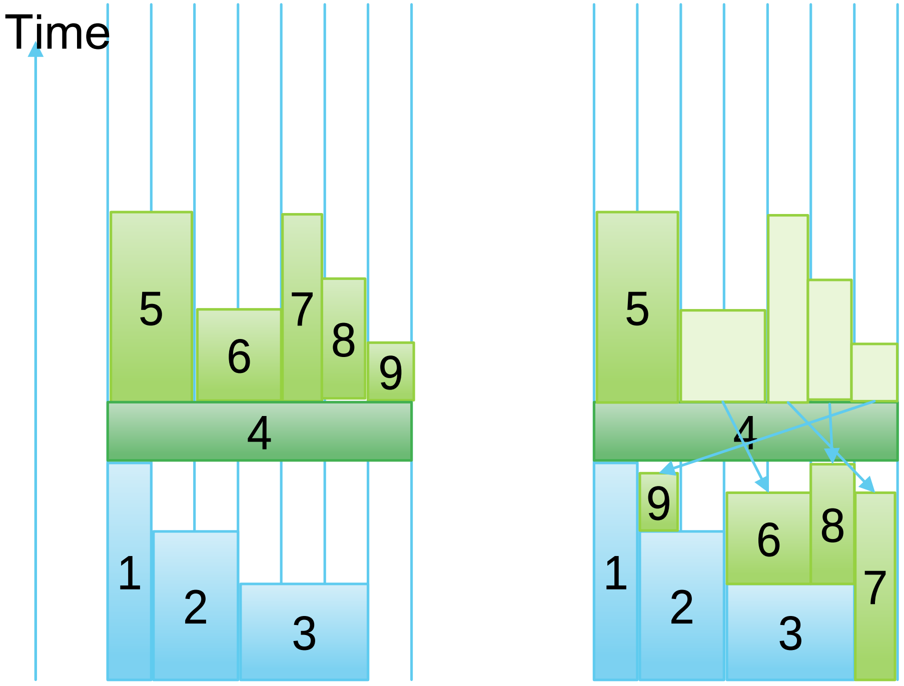
System-level monitoring environment

- The date is open to everyone!
  - http://mon.g.gsic.titech.ac.jp/
  - Not limited to TSUBAME users and administrators
  - Sometimes used as the basic data of research (FT, scheduler, …)
    - Contact us if the data on web is insufficient for you
- We also have log data of batch queue
  - Used for accounting
  - Cannot be disclosed because it contain lots of users’ privacy
    - TSUBAME2.5 is used by industrial users as well as academia
- Those data should be used for ***optimization*** done ***for system***
  - We have never analyzed ***quantitatively***…



Case study
Let's optimize users behavior!

- Batch queue scheduler with backfill does not work if users don't predict their execution time correctly



w/o Backfilling   with Backfilling

Scheduler can fill smaller jobs if it finishes earlier than start time of bigger job

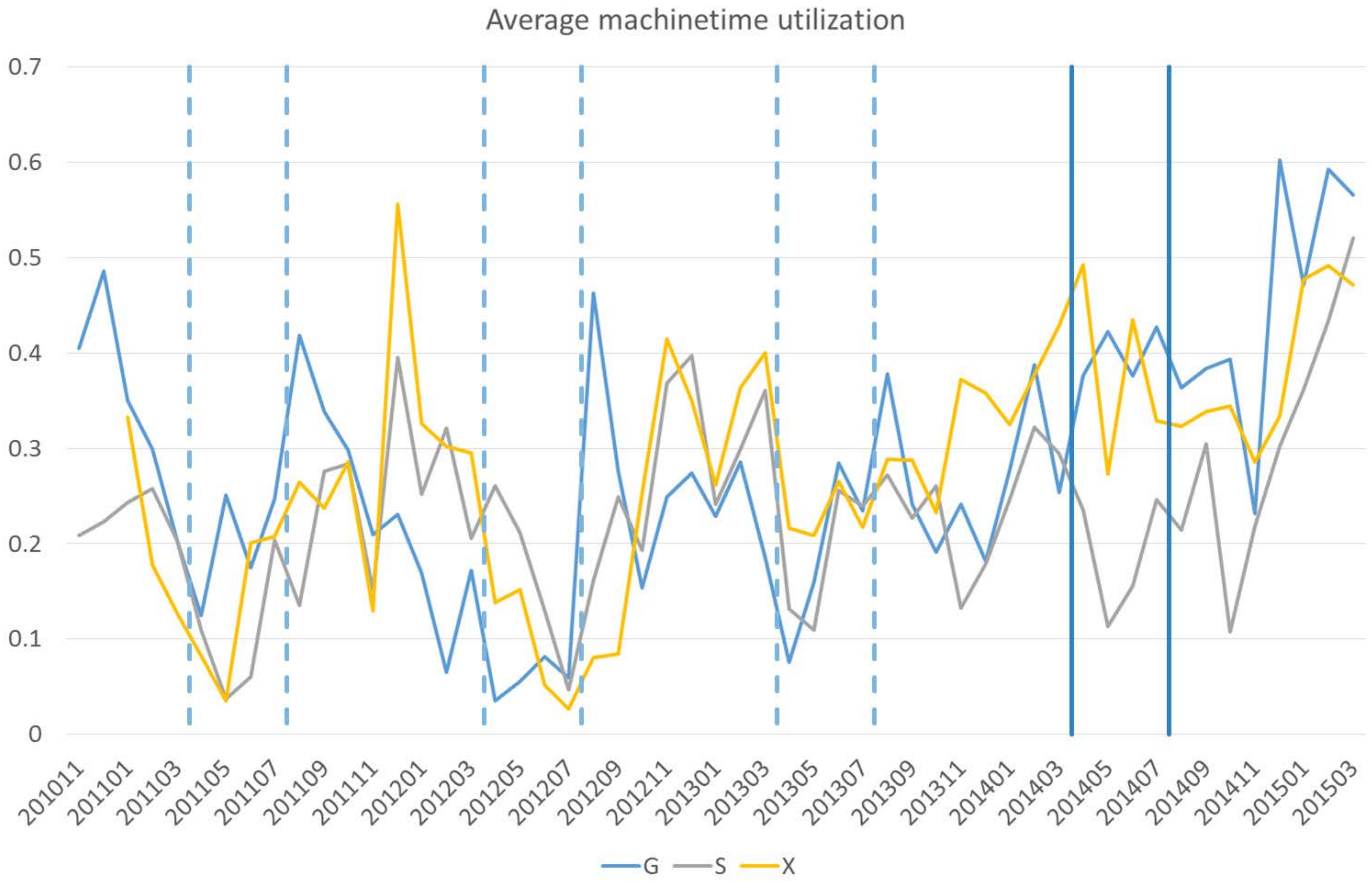
This calculation is done over ***estimated*** execution time, not actual execution time

Solution: Giving monetary incentives to users specifying accurate execution time

- Prior to April 2014, lots of users specified execution time as 24 hour
  - No explicit incentive to specify shorter than 24hr
- We ***charge less*** (× 0.9) when user specify execution time < 1 hour from April 2014
- We started charging to ***specified*** execution time as well as actual execution time at August 2014
  - (1 × actual time) → (0.7 × actual time + 0.1 × specified time)
  - We charge ***more*** if user specify ×3 execution time or more
  - We charge ***less*** otherwise
- To verify the effect, let's check (actual/specified) execution time ratio

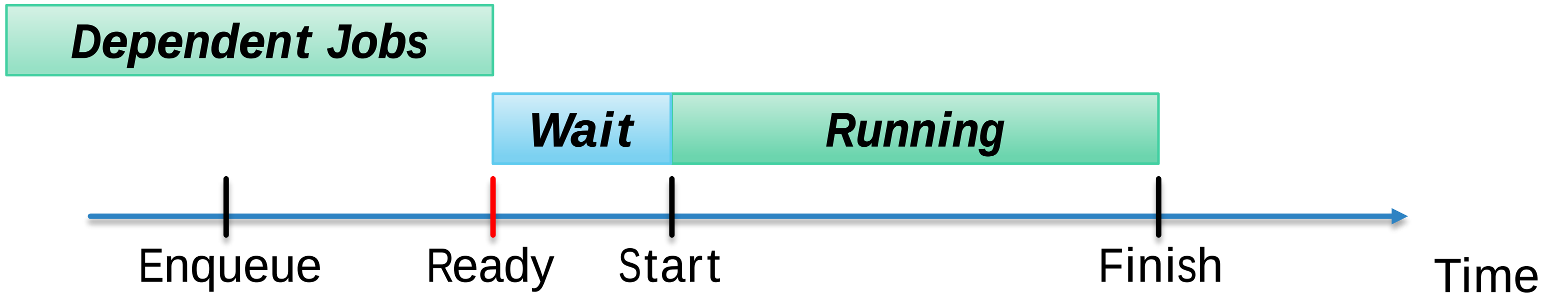
Execution time (actual/specified) ratio

- User started specifying more accurate time after accounting system changed
- The effect is different among the job classes
  - Better change in more restrictive queues (exec time, GPU requisite)
  - Advanced users tend to adopt much
- Note: we sometimes reach different conclusion in many reasons
  - Choice of metrics
  - Other factor affects the result: *Thesis season!*



What we really wanted to verify…

- The original goal was “better usage in batch queue”
- We should have checked the average turnaround time for jobs



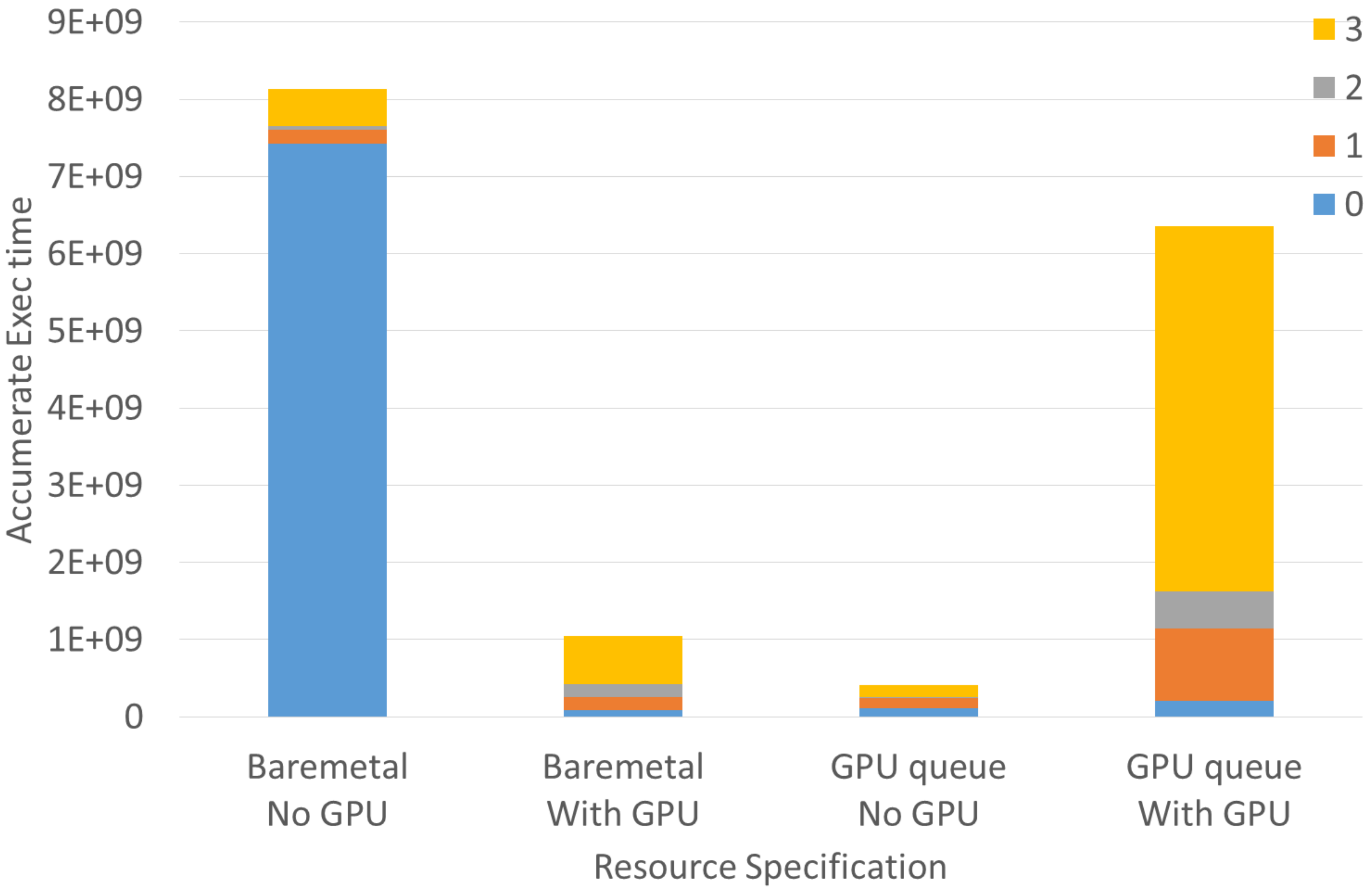
- But we didn’t have timing data of the time job became ready to run
  - We didn’t record the job dependency… This must be future work for next system

Case study
How much users lying to scheduler?

- Some resource specification to scheduler is not used for actual resource reservation
  - Memory: scheduler kills the job which exceeds the specification
    - User specifies accurately
  - #GPUs: scheduler does nothing about GPU usage
    - User may tell lie
- We shouldn’t use this specification for statistics for design of next-gen machine
- Let’s verify how much users telling lie
  - We have GPU monitoring log and scheduler log

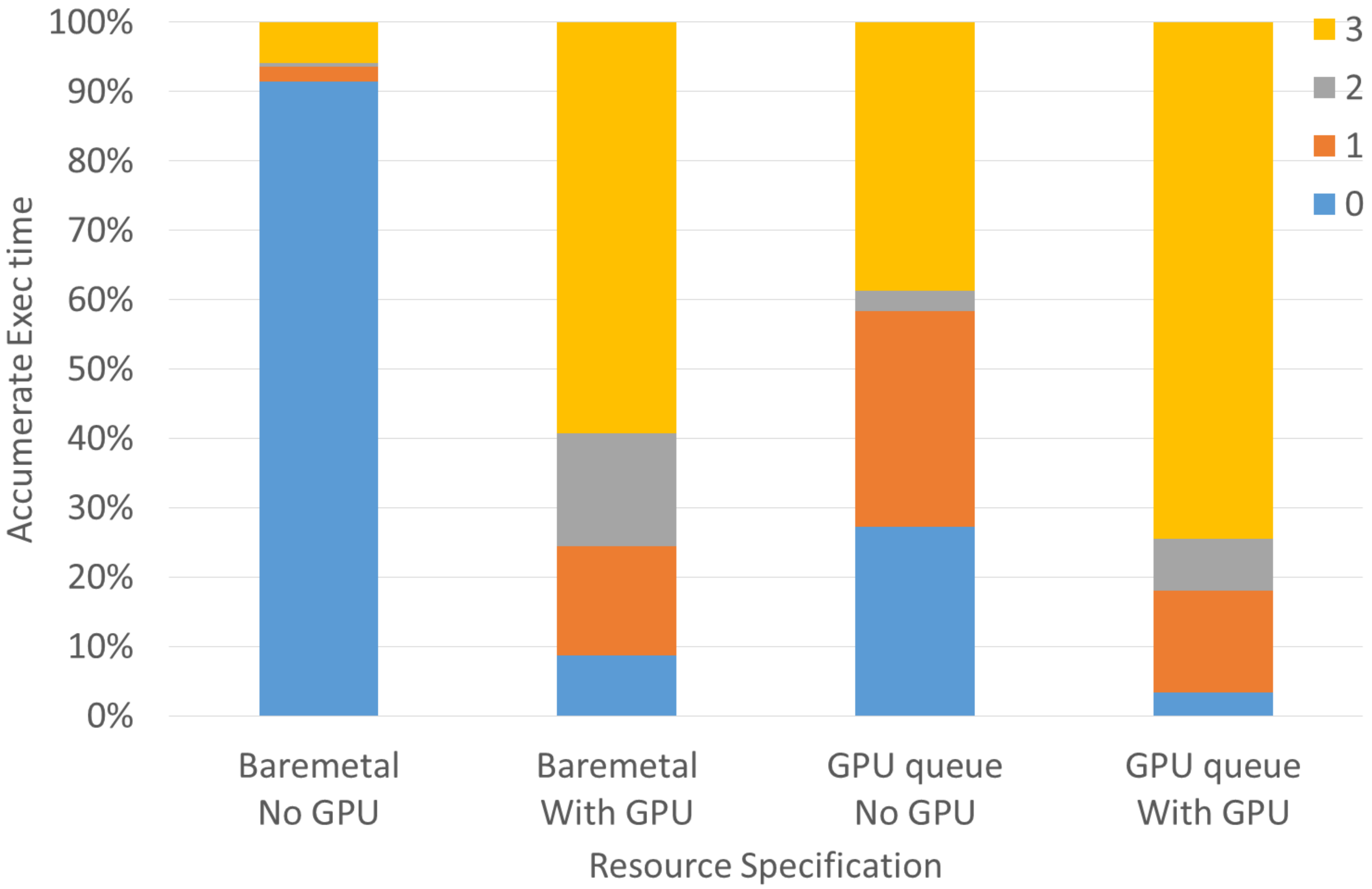
Result: How much user telling lie?

- ~10% Users telling lie!
- Some users sending no GPU jobs to GPU queue…



Result: How much user telling lie?

- ~10% Users telling lie!
- Some users sending no GPU jobs to GPU queue…



Summary

- Performance analysis and monitoring is crucial not only for supercomputer user but also for service providers
  - Better performance of user app increases effective resources provided
  - System level monitoring logs are often forgotten, but they contains lots of treasures to understand the computer's usage well, which leads better computer design in future.
- Future work
  - More analysis on the data we have
    - What is typical bottleneck in our machine?
    - Suggestion of analysis is very welcome
  - Share the data with others, collect data as much as possible
    - Utilization data is often concealed (at least in Japan) ☹
    - Common data format?
    - What metric should we start collecting in next system