Making
Commercially-Sensitive Workloads
Safe to Release

Charles Reiss (UC Berkeley)
John Wilkes (Google)
Joseph L. Hellerstein (Google)
Motivation

We had a *cluster scheduler* trace

- *Machines* and their availability
- *Jobs* submitted by *users* composed of many *tasks* (request to run a VM)
- Resource *requests* and *usage* (per task)
- When/where each task was started and stopped

Enables new research:

- Batch + interactive together
- Research needs realistic example
Obstacles to releasing traces

Privacy

Competitive concerns
Privacy

"information ... that can be used to contact or identify [any user]"

Regulatory restrictions

Subject of most prior work:
  – (De)anonymizing Netflix prize
  – Researching health info legally

Quote: Quantcast privacy policy
Competitive concerns

"could be used to hurt the company"

• Info sensitive for X, Inc. may be in press releases for Y, Inc.
• Examples:
  – Performance/capability numbers
    • Competitors could compare
  – Supply chain issues:
    • Committed to certain devices
Goal

Select a version of the raw workload that is safe to release

*Obfuscating* the workload
Why is obfuscation hard?

• **Outside** data sources
  – Zip code + Date of Birth

• Some **aggregates** are sensitive
  – Total number of users, machines, etc.
Obfuscation techniques

• Transform
  – preserve equality/order/etc.

• Subset
  – representative, not complete

• Aggregate
  – provide only summaries
Transforming

• *Choose* what users need to do
  – "check equality"
  – "check if less/greater”

• Each datatype independently
• Choose parameters neutrally
  – e.g. maximum value becomes 1; not "random" scaling factor
Example: Task constraints

e.g. "foo_version >= 143"

Specify machines that can run task

Allow
• comparing attribute values *only*
Example: Task constraints

Solution:
- foo_version becomes MAC(secret, foo_version)
  - secret only used for this purpose

- For each attribute:
  - sort values that actually appear
  - rename values 1, 2, 3, ...
Example: CPU usage

Can we allow:

• Summing CPU usages and comparing to capacities
  — effectively requires linear scaling

but not allow?

• discovery of machine core counts
Example: CPU usage
Example: CPU usage
Example: CPU usage

No transformation that allows summing usages will avoid revealing "1 core"

Compromise:

• Choose *subset of machines* for which revealing core count is okay
• *representative* of the workload type
Example: Job purpose

Applications with different performance goals

Researchers want the **semantics** of jobs

*Internally*: job names + user names

*Manually* label **1000s** of job names??
Example: Job purpose

Compromise:
• Scheduler parameters
  – priority, latency-sensitivity
• Extra measurements
  – CPI, memory traffic

... but no way to verify purpose
Aggregation

• Summaries only
  – e.g. 5/25/50/75/95th percentiles

• Good for privacy

• But need to choose what's important
  – We didn't really know
Conclusion

• Releasing useful traces is hard
• Privacy isn't enough
• Be systematic
  – Choose what trace users should do
• Subsetting often more useful than field-by-field transformations

• No free lunch ... but we got a trace.
We released a trace at [https://code.google.com/p/googleclusterdata/](https://code.google.com/p/googleclusterdata/).