An Empirical Study on Configuration Errors in Commercial and Open Source Systems

Yin, Z., Ma, X., Zheng, J., Zhou, Y., Bairavasundaram, L. N., & Pasupathy, In *Proceedings of SOSP 2011*.

Presented by: Ala’ Alkhaldi

April 16, 2013

Configuration errors

- Very Significant
  - Great impact on system availability: e.g. Facebook outage, 2010
  - Prevalent: 27%-50% of system faults
  - Very expensive: Technical support costs 17% of the total systems cost
- Difficult to study
  - Poorly documented: Undetailed issue-repositories, and in the form of unstructured user driven textual descriptions
  - Confidential information
  - Studying them is mostly a manual task !

Research efforts

- Detection

  PeerPressure uses statistical methods on large configuration sets to identify single configuration parameter errors

- Diagnosis

  AutoBash tries out fixes from a solution database to find proper solution to a configuration problem

- Avoidance

  Using predefined rules (SmartForg), machine learning, or templates for automatic configuration generation

- Tolerance and online validation

This paper studies the characteristics of the real-world configuration errors

- Statistics and classifications would benefit current research directions and tools.
- Guiding system developers in designing systems configuration logic

Data sets

- Sources
  - COMP-A storage system
    - Closed cases in the customer-issue DB
    - 1000 cases marked as “Configuration” are filtered to be 309 cases
  - Open-source systems
    - Closed cases from official support forms, mailing lists, and ServerFaults.com
    - 237 cases are randomly sampled
  - Data is manually processed

System	Total Cases	Sampled Cases	Used Cases
COMP-A	confidential	1000	309
CentOS	4338	521	60
MySQL	3340	720	55
Apache	8513	616	60
OpenLDAP	1447	472	62
Total	N/A	3329	546

- Concerns about data validity and limitations
  - Data set size and sampling error
  - Relying on user reported error
    - Trivial errors are not reported.
    - Expert vs. novice users
  - Configuration error fixed by user environments are not considered
  - It doesn’t differentiate between system versions

The study covers

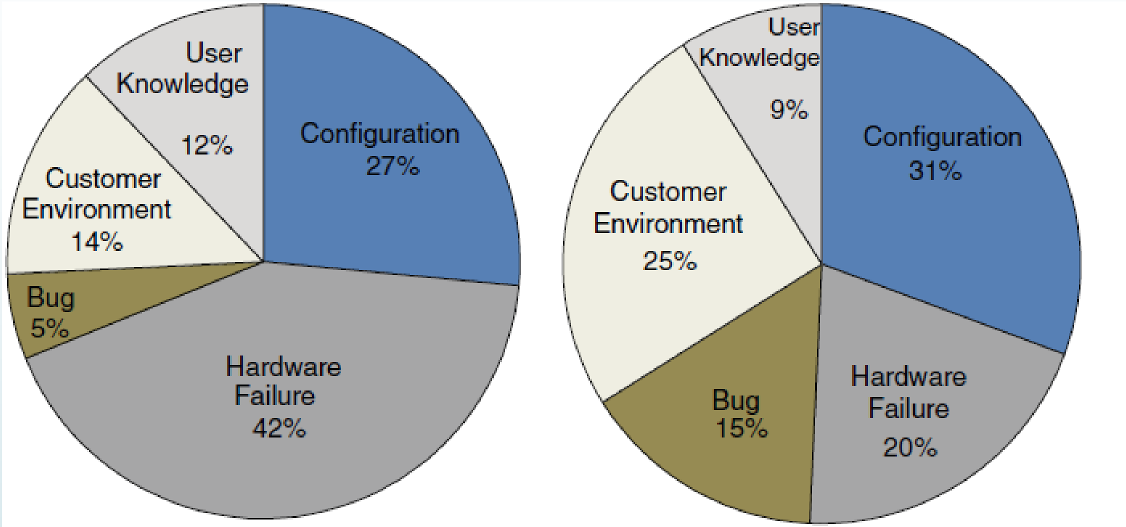
1. Prevalence of configuration errors
2. Types of configuration errors
3. System reactions to configuration errors
4. Frequency of different causes of configuration errors
5. Impact of configuration errors

Warning: There will be lots of graphs and numbers. Bear with me!

1. Prevalence of configuration errors

Finding 1:
A significant percentage of customer cases are related to configuration issues.

Finding 2:
Configuration issues causes the largest percentage of high-severity support requests.



(a) Categorization of problem causes on all the cases (b) Categorization of problem causes on cases with high severity

- Problem cause and severity are identified by COMP-A engineers
- No such labeled data is available for open source systems
- Configuration percentage might be inflated by the popularity of customer requests for conf. information

2. Types of configuration errors

1. Configuration Parameter mistakes

2. Software compatibility errors

3. Other errors (Component)

System	Parameter	Compatibility	Component	Total
COMP-A	246 (79.6±2.4%)	31 (10.0±1.8%)	32 (10.4±1.8%)	309
CentOS	42 (70.0±3.7%)	11 (18.3±3.1%)	7 (11.7±2.6%)	60
MySQL	47 (85.5±2.3%)	0	8 (14.5±2.3%)	55
Apache	50 (83.4±2.8%)	5 (8.3±2.1%)	5 (8.3±2.1%)	60
OpenLDAP	49 (79.0±3.0%)	7 (11.2±2.3%)	6 (9.7±2.2%)	62

Sampling error

2.1. Parameter configuration errors

- Parameter: is a value set in a configuration file or sent through a console command
- 70%- 85% of configuration mistakes
  - Raises a flag for system designers to create less config-knobs and more auto-config
  - Can be detected automatically by checking against configuration rules
- Two Types:
  - Legal (46%-62%): Syntactically correct but causes functional and performance problems **(Hard to detect)**
  - Illegal:
    - Illegal format: Lower/upper case, field separator, etc
    - Illegal values (The majority): parameter value violates some constraint or inconsistent with other values or with the environment

(a) Illegal 1 – Format – Lexical from COMP-A

InitiatorName: **iqn:DEV_domain**

Description: for COMP-A's iscsi device, the name of initiator (**InitiatorName**) can only allow lowercase letters, while the user set the name with some capital letters "DEV".

Impact: a storage share cannot be recognized.

(c) Illegal 3 – Format – Syntax from Apache with PHP

extension = mysql.so
……
extension = recode.so

"recode.so" must be put before "mysql.so"

Description: When using PHP in Apache, the extension **"mysql.so"** depends on **"recode.so"**. Therefore the order between them matters. The user configured the order in a wrong way.

Impact: Apache cannot start due to seg fault.

(e) Illegal 5 – Value – Env Inconsistency from COMP-A

192.168.x.x system-e0

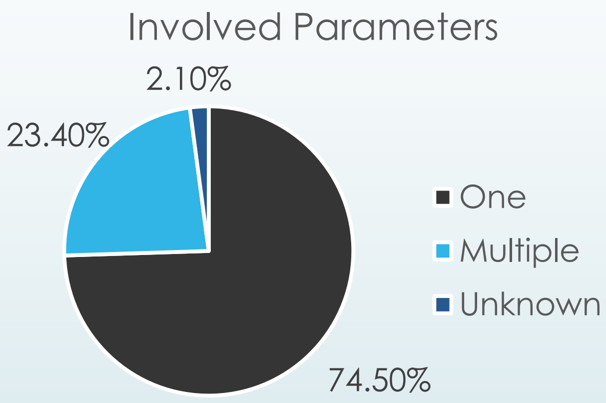
There is no interface named "system-e0"

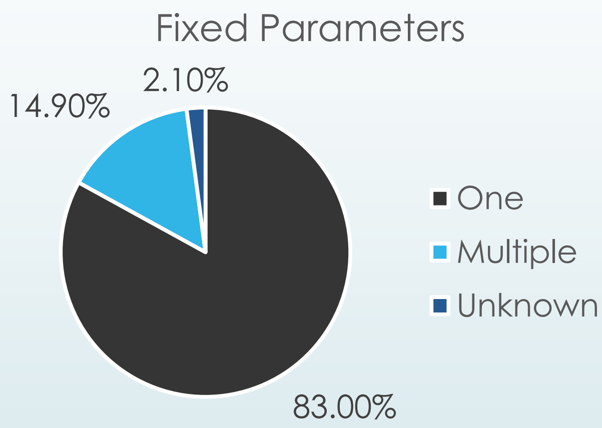
Description: In the hosts file of COMP-A's system, The mapping from ip address to interface name needs to be specified. However, the user mapped the ip **"192.168.x.x"** to a non-existed interface **"system-e0"**.

Impact: The host cannot be accessed.

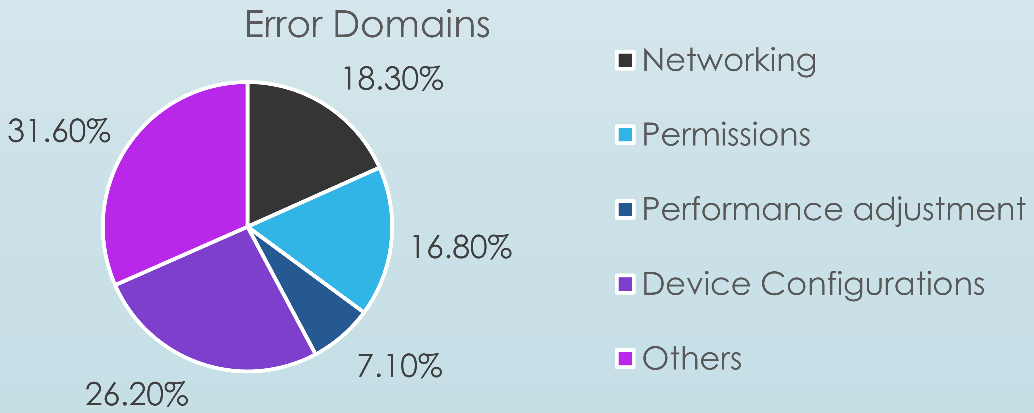
2.1. Parameter configuration errors, cont.

- Number of Erroneous parameters





- Problem domains of parameter mistakes



2.2. Software compatibility configuration errors

- Improper combinations of components or their version.
  - 18.3% of configuration error types
- Software upgrades are not a major source of these errors (only 18.5%)
- Could be mitigated by using package-management systems, self-contained packages, or delivering the system as virtual machine

2.3. Component configuration errors

- Errors related to how the system is organized and how resources are supplied, e.g.: missing software components, error in files format, etc.

Subtype	Number of Cases
Missing component	15(25.9%)
Placement	13(22.4%)
File format	3(5.2%)
Insufficient resource	15(25.7%)
Stale data	3(5.2%)
Others	9(15.5%)

3. System reaction to configuration errors

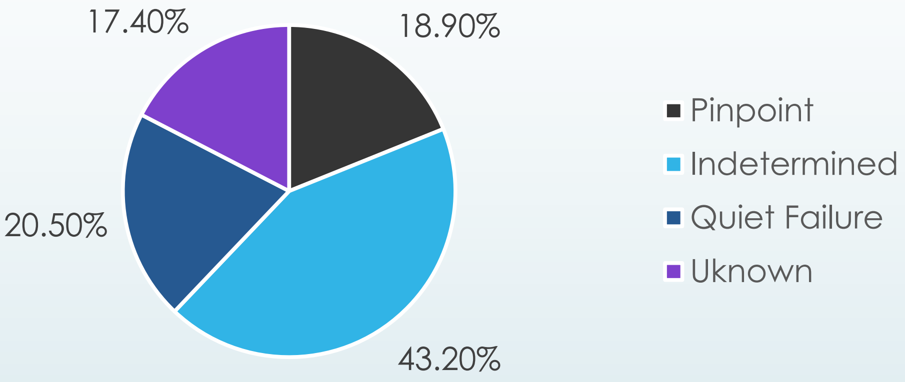
System	Pinpoint Reaction	Indeterminate Reaction	Quiet Failure	Unknown
COMP-A	48(15.5±2.2%)	153(49.5±3.0%)	74(23.9±2.6%)	34(11.0±1.9%)
CentOS	7(11.7±2.4%)	33(55.0±3.7%)	16(26.7±3.3%)	4(6.7±1.9%)
MySQL	4(7.2±1.7%)	26(47.3±3.2%)	13(23.6±2.8%)	12(21.8±2.7%)
Apache	8(13.3±2.6%)	28(46.7±3.8%)	16(26.7±3.4%)	8(13.3±2.6%)
OpenLDAP	9(14.5±2.6%)	28(45.2%±3.7%)	14(22.6±3.1%)	11(17.7±2.8%)

Finding: Only 7%-15% of the studied configuration provides explicit messages that pinpoint the problem configuration error

- Quiet Failures could cause mysterious behaviors
  - Example: A web application used both mod_python and mod_wsgi modules in Apache server. These two modules used two different version of Python, which caused segmentation fault errors when trying to access the web page.
  - 5%-8% of the cases

3. System reaction to configuration errors, cont.

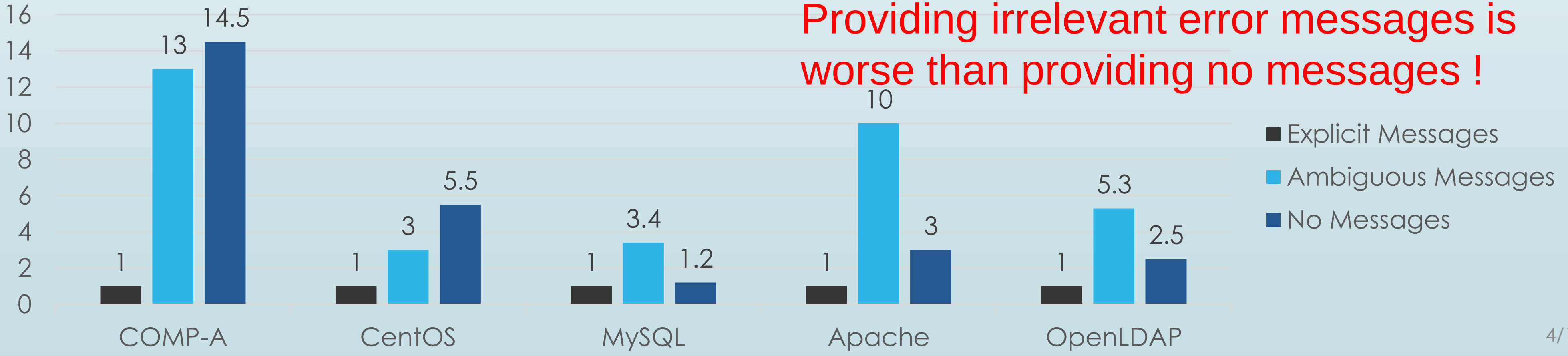
COMP-A Reactions to Illegal Parameters



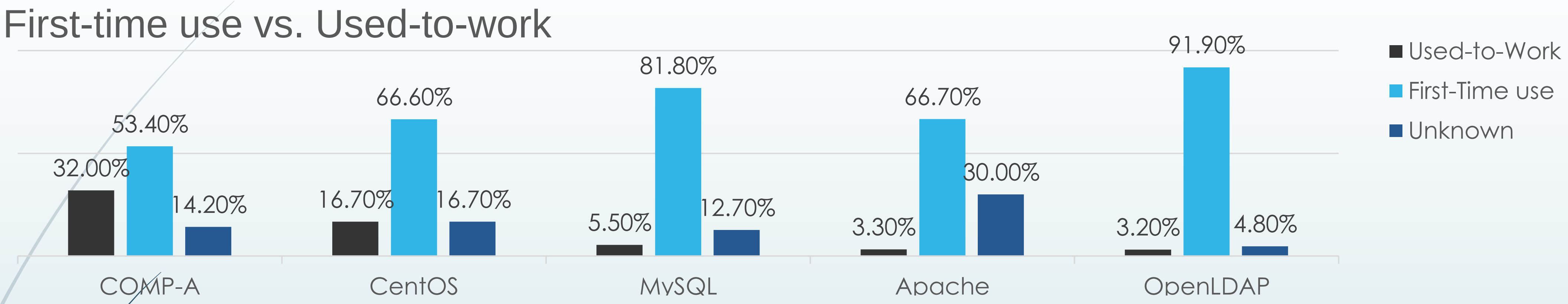
Although illegal parameters are easier to check, only 18.90% of reactions pinpoint the actual problem !

Normalized Median of Diagnosis Time

Providing irrelevant error messages is worse than providing no messages !



4. Causes of configuration errors

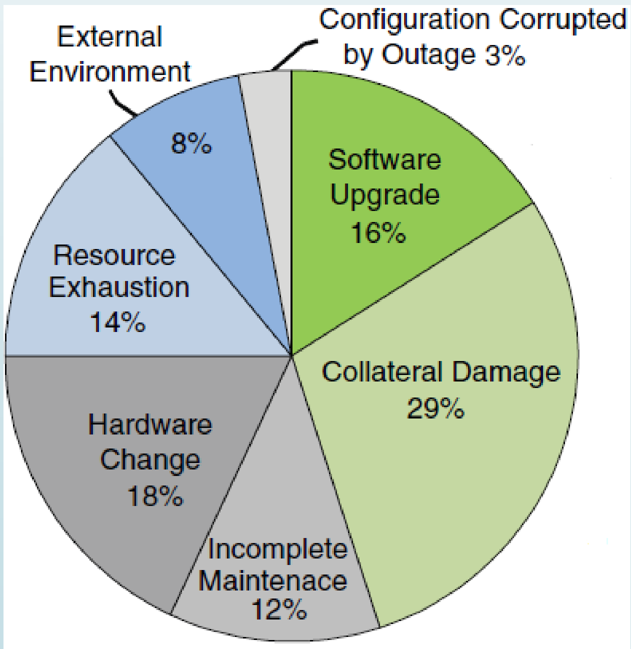


Finding1: First-Time use errors are the majority. The reasons are: [1] Lack of knowledge [2] Flawed system design [3] Inconsistent manuals

Finding2: In complex systems (COMP-A & CentOS) the frequency of system changes and the complexity of configuration increases the probability of used-to-work configuration errors

Why do systems stop working in Used-to-Work cases?

Finding3: Parameter-related configuration errors (Collateral damages, incomplete maintenance, and configuration corrupted by outage) can benefit from tracking configuration changes and validation.



The cause distribution for 100 case of used-to-work configuration error at COMP-A

5. Impact of configuration errors

System	Fully Unavailable	Partially Unavailable	Performance Degradation
COMP-A	41 (13.3±2.1%)	247 (79.9±2.4%)	21 (6.8±1.5%)
CentOS	12 (20.0±3.2%)	47 (78.3±3.3%)	1 (1.7±1.0%)
MySQL	15 (27.3±2.9%)	29 (52.7±3.2%)	11 (20.0±2.6%)
Apache	15 (25.0±3.3%)	44 (73.3±3.4%)	1 (1.7±1.0%)
OpenLDAP	6 (9.7±2.2%)	52 (83.9±2.7%)	4 (6.4±1.8%)

- The performance of database systems is very sensitive to configuration errors
  - In most cases database performance tuning manuals have hundreds of configuration parameters !

Misconfig Type	Fully Unavailable	Partially Unavailable	Performance Degradation
Parameters	59 (**13.6%**)	342 (78.8%)	33 (7.6%)
Compatibility	14 (**25.9%**)	38 (70.4%)	2 (3.7%)
Component	16 (**27.6%**)	39 (67.2%)	3 (5.2%)

- Compatibility and component errors have severe impact and harder to fix.

Summary

- Configuration errors are significant and could lead to sever consequences.
- Parameter configuration errors represents a majority of error types.
  - They can be avoided and easily fixed.
- Configuration options should be as minimal as possible by design.
  - Using Auto-configuration and ready to use software is a mitigation.
- In case of configuration errors the system should react in details and pinpoint the root causes of the problem.

Discussion

- Trade of between flexibility in working options and avoiding configuration problems
- Handling configuration errors in open source systems vs. commercial systems.
  - Availability of support service and customer-issue databases.
- Without proposition of suggestions or possible solutions, this paper is more like a technical white paper than an academic one.

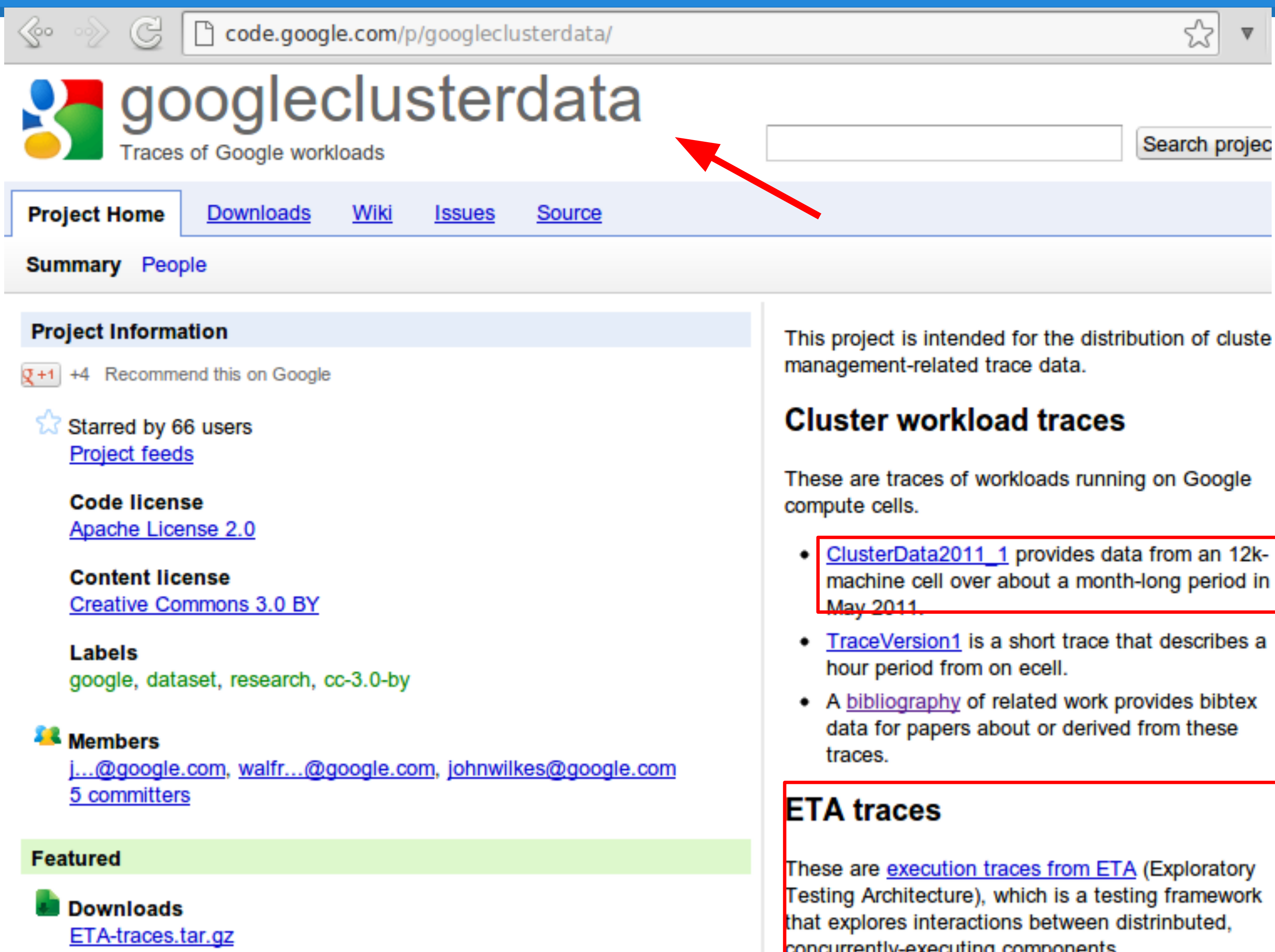
SO Much data...

Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis

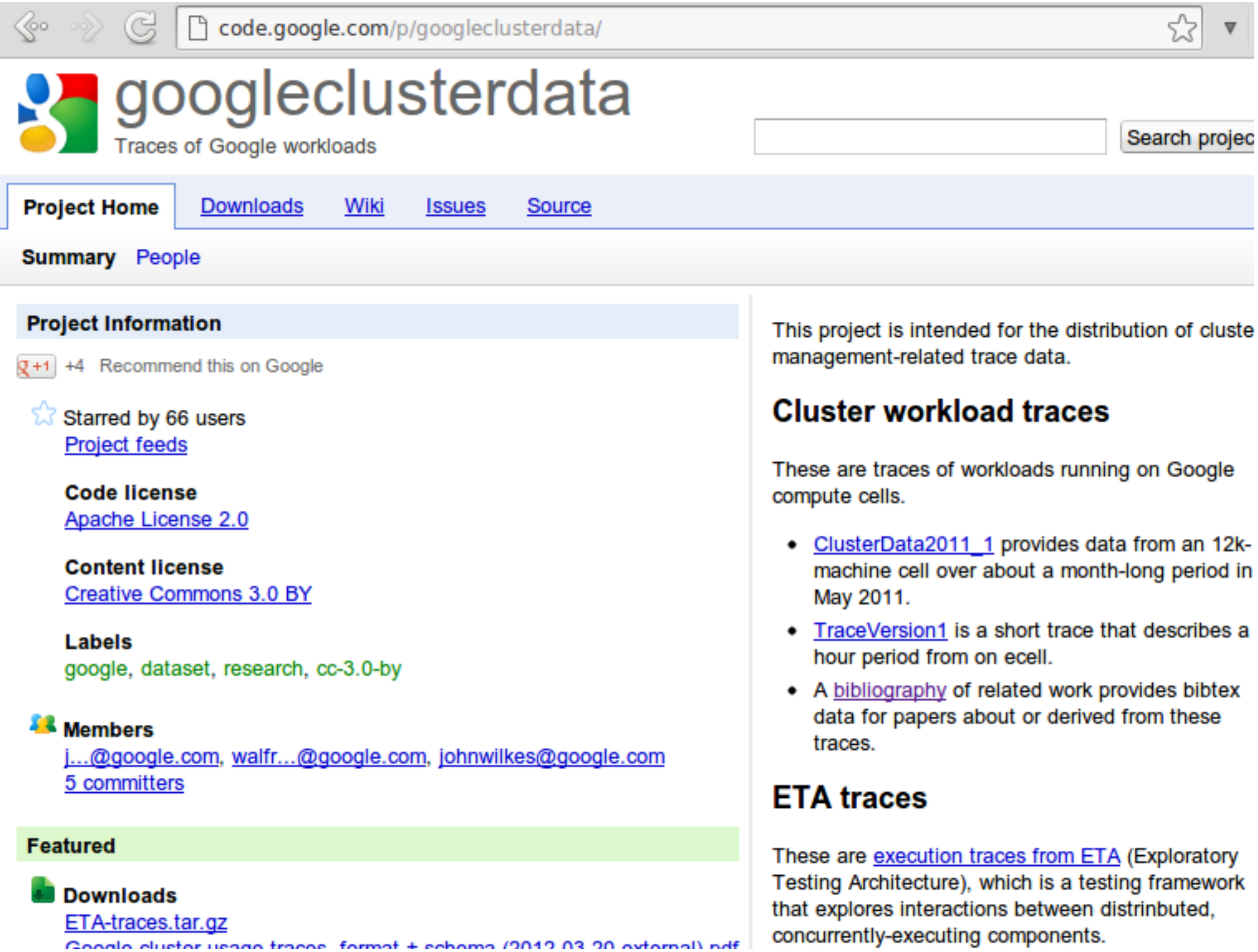
Presented by Faraz Faghri

*material is taken from the paper and slides.

Story begins ...



Google cloud cluster



- Large-scale
- Multi-purpose
- Heterogeneous
    - Hardware
    - Job demand

Data ...

- Data from cluster scheduler.

Tasks (25M): ‘run a program somewhere once’: more like MapReduce worker than MR task

Jobs (650k): collections of related tasks. no formal co-scheduling requirement.

- 12.5K machines, one month.

Number of machines	Platform	CPUs	Memory
6732	B	0.50	0.50
3863	B	0.50	0.25
1001	B	0.50	0.75
795	C	1.00	1.00
126	A	0.25	0.25
52	B	0.50	0.12
5	B	0.50	0.03
5	B	0.50	0.97
3	C	1.00	0.50
1	B	0.50	0.06

Lessons to be learned ...

For effective cloud-based schedulers.

Google cluster properties:

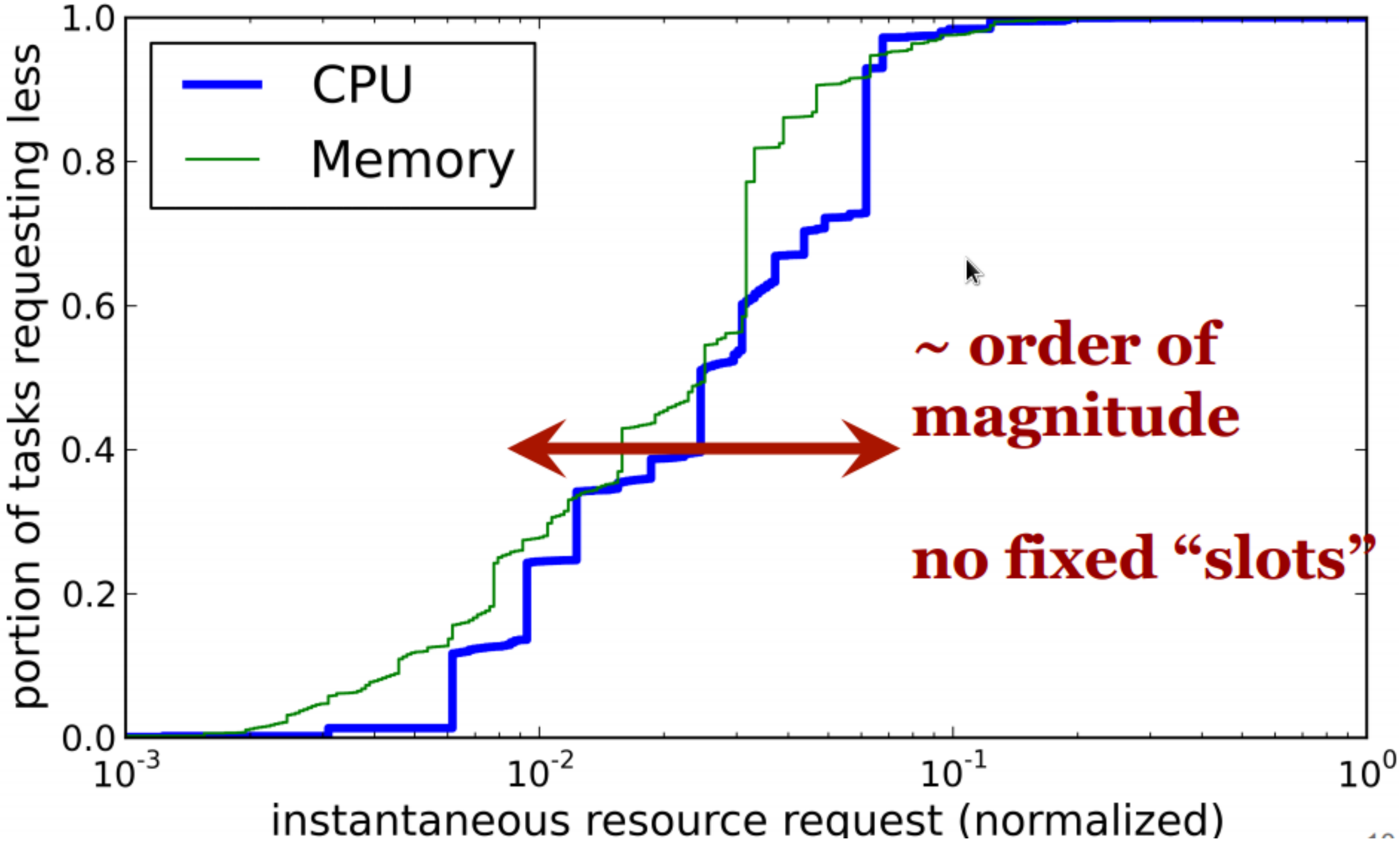
!!!

- Run all workloads on one cluster!
    - Increased efficiency:
        Fill in “gaps” in interactive workload Delay batch if interactive demand spikes.
    - Increased flexibility:
        Share data between batch and interactive.

Variety of workloads: may be multiple clusters?

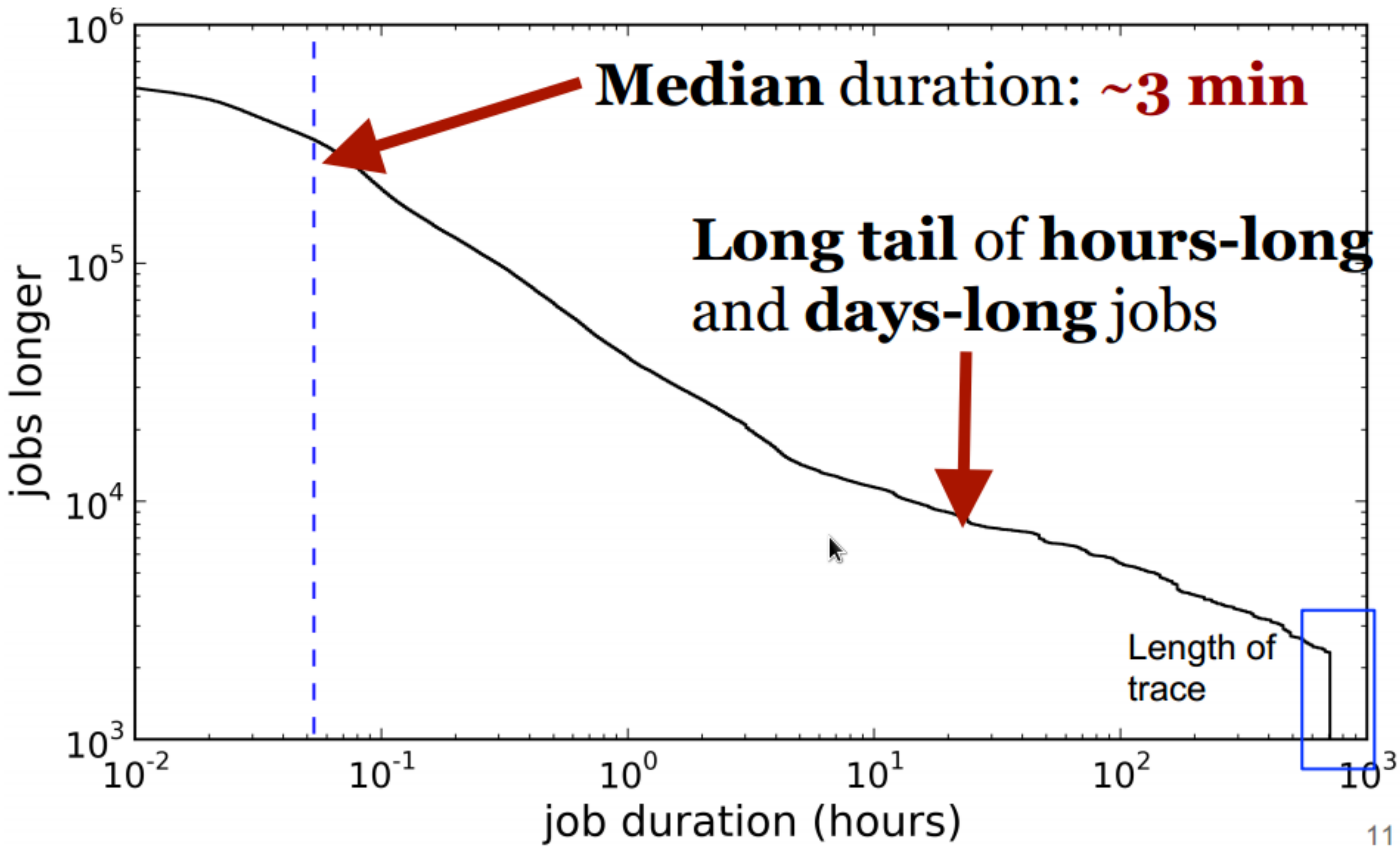
Traditional schedulers assumptions:

- Tasks could be slot and core based.



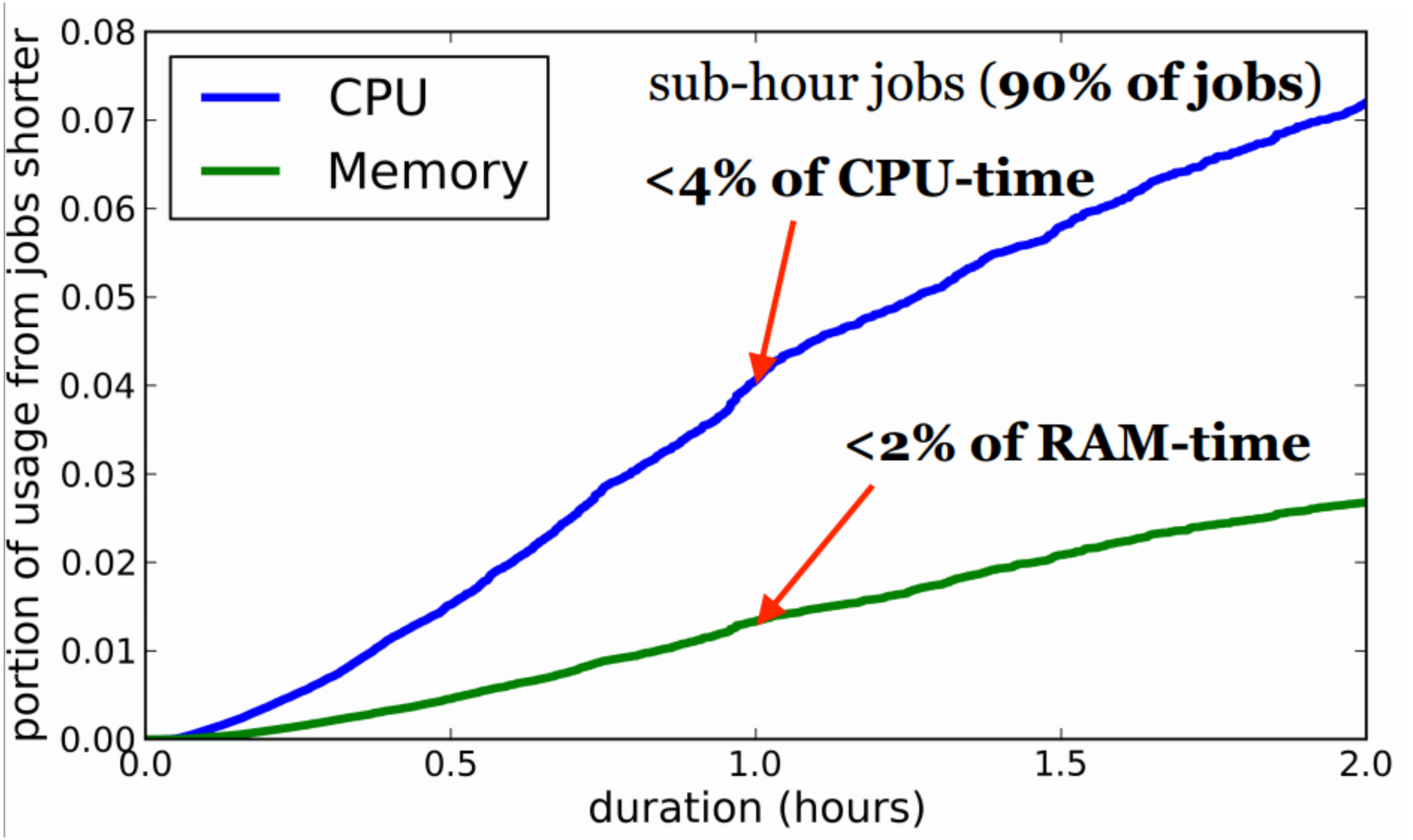
Traditional schedulers assumptions:

- low variety of workloads (time variant).



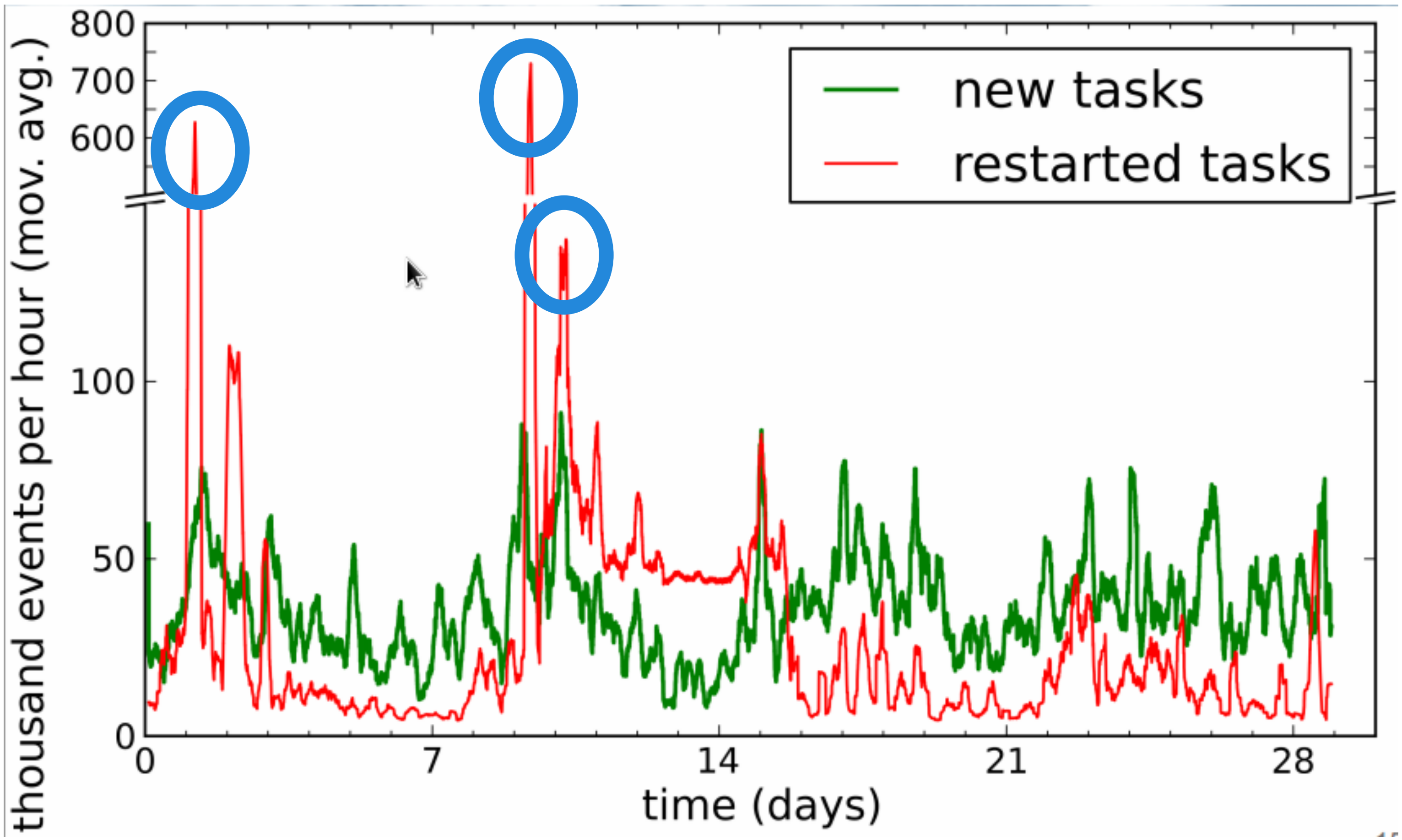
Traditional schedulers assumptions:

- Long-running tasks are most usage.



Traditional schedulers assumptions:

- Schedulers don't need to act very frequently.
- 100K+ decisions per hour.



Traditional schedulers assumptions:

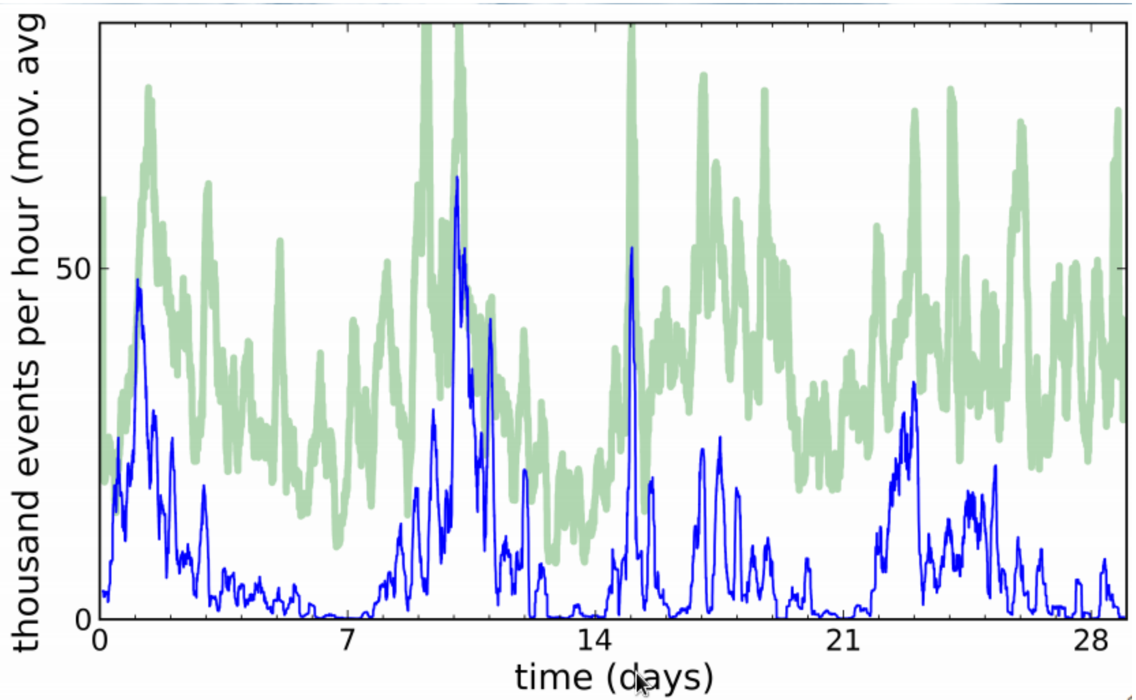
Evictions of **higher-priority tasks** and **machine downtime:**

- Coincide with those tasks starting:

    0.04 evictions/task-hour for lowest priority.

- 40% of machines down once in the month:

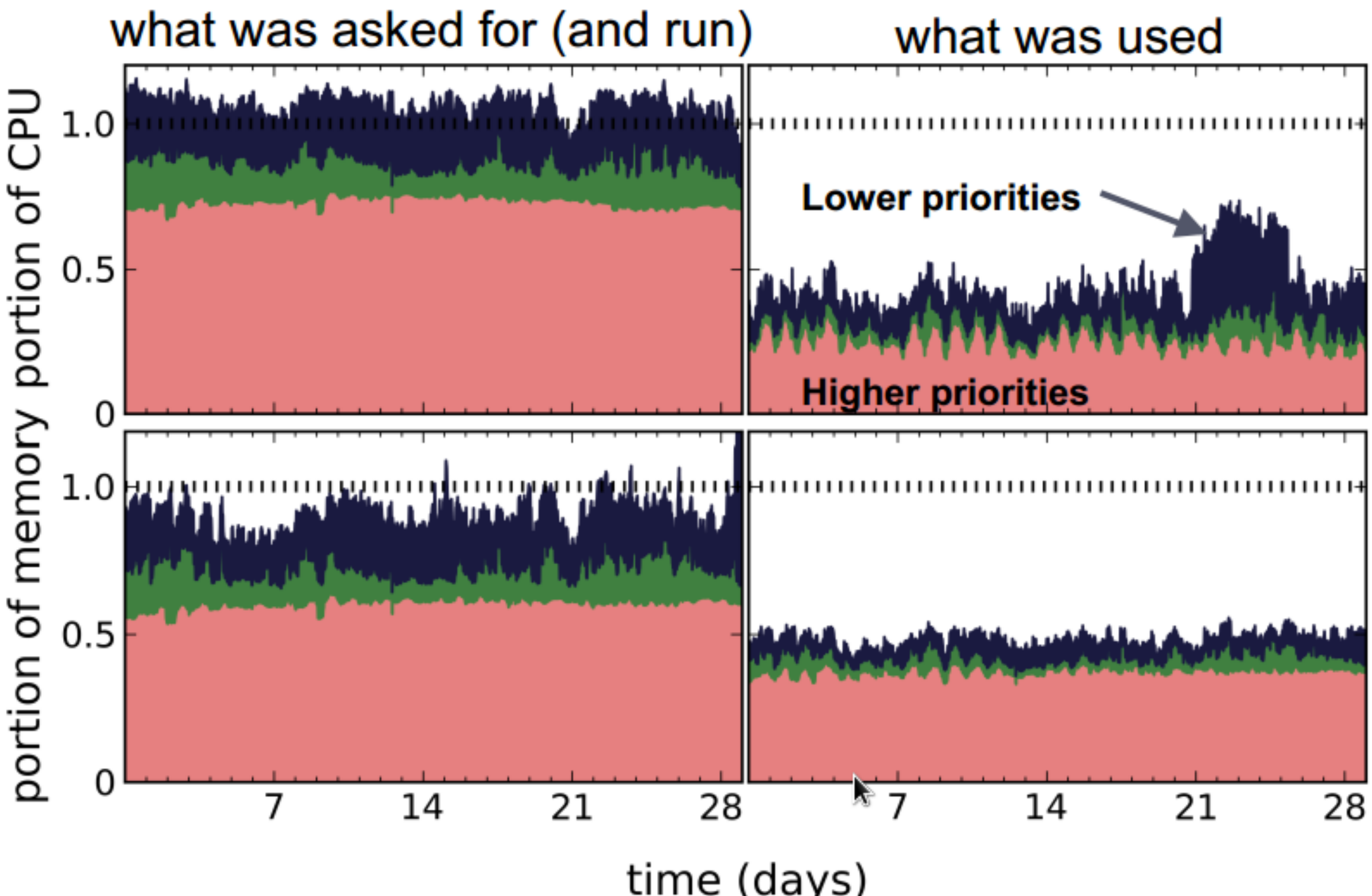
    Upgrades, repairs, failures.



Traditional schedulers assumptions:

- We have resource estimations and we can trust them.

Wstimate **worst-case usage**: **~60%** of difference from average usage



Traditional schedulers assumptions:

- Machines are homogenous.

Tasks can restrict acceptable machines (for reasons other than resources)

Used by ~6% of tasks

Examples: Some jobs require each task to be on a different machine

Number of machines	Platform	CPUs	Memory
6732	B	0.50	0.50
3863	B	0.50	0.25
1001	B	0.50	0.75
795	C	1.00	1.00
126	A	0.25	0.25
52	B	0.50	0.12
5	B	0.50	0.03
5	B	0.50	0.97
3	C	1.00	0.50
1	B	0.50	0.06

Call For Schedulers!

- Complex workloads.
- Complex task requests.
- Complex resources.
- Complex task constraints.
- Distributions not match a power law, lognormal, Weibull, or exponential distribution.

Operation Research folks have worked on that.

Don't forget!

- Rapid scheduling decisions.
- Complex task restarts.
- No reliable estimations given from tasks.
- Central scheduler might not work, lot's of immediate changes across a BIG cluster.

Discussion

- How representative is **Google cluster** and **Google traces**?
- **Why** to have such a multi-purpose cluster?

- Should we go and design a scheduler with this data, how **valid** are these numbers with new scheduler?

- **Piazza**:The authors postulate that the resource requests are being specified manually. Using machine learning techniques, this should be feasible to be performed for more efficient usage of resources.