ORACLE

What we talk about when we talk about scheduling

Tim Harris
Oracle Labs

The following is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.

How many schedulers do we need?

Analytics workload running on a shared cluster

How many schedulers do we need?

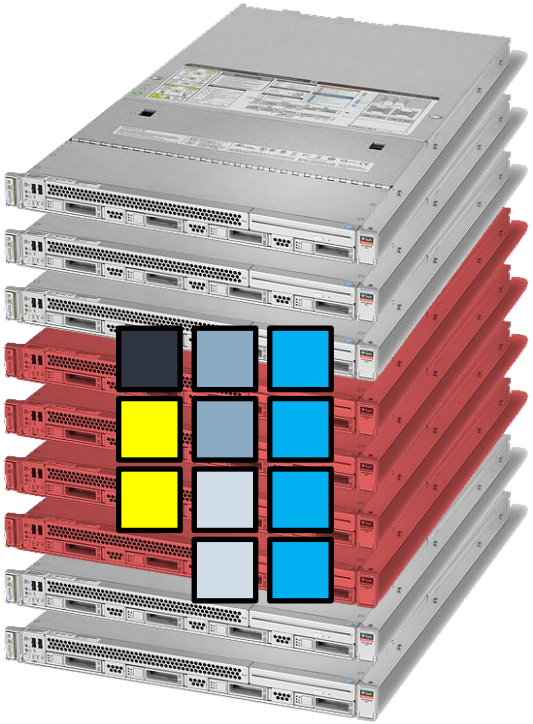
Analytics workload running on a shared cluster

1. Long-term allocation of a set of machines using Slurm. Run Yarn within those.

How many schedulers do we need?

Analytics workload running on a shared cluster

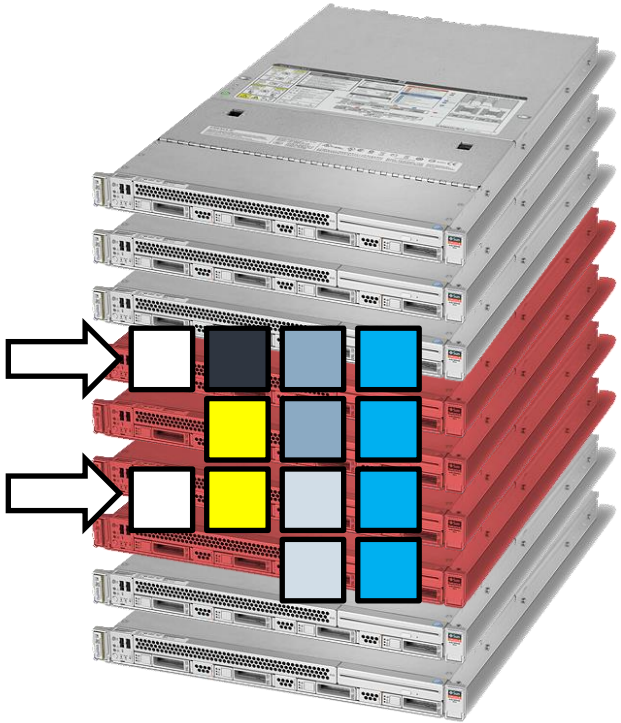
2. Start job running within Yarn – let's assume it just needs to run on two machines.



How many schedulers do we need?

Analytics workload running on a shared cluster

2. Start job running within Yarn – let's assume it just needs to run on two machines.



How many schedulers do we need?

Analytics workload running on a shared cluster

3. Load data into memory. Contention with other jobs accessing HDFS.

How many schedulers do we need?

Analytics workload running on a shared cluster

3. Load data into memory. Contention with other jobs accessing HDFS.

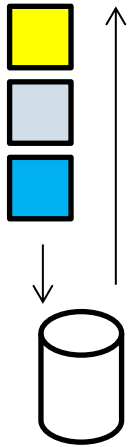
- Which replica to access?

How many schedulers do we need?

Analytics workload running on a shared cluster

3. Load data into memory. Contention with other jobs accessing HDFS.

- Which replica to access?
- Which order should that replica service requests?



How many schedulers do we need?

Analytics workload running on a shared cluster

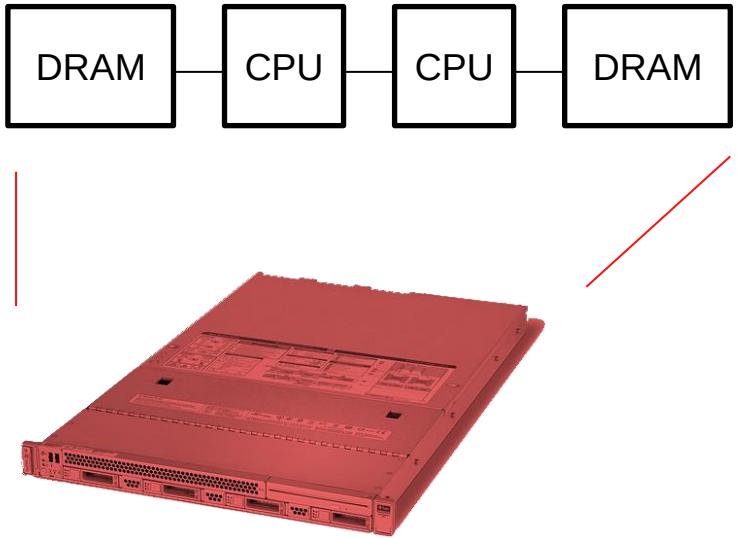
4. Within a query, how to distribute tasks between the machines involved.

- If using work-stealing, who to steal from? How important is it to preserve locality?

How many schedulers do we need?

Analytics workload running on a shared cluster

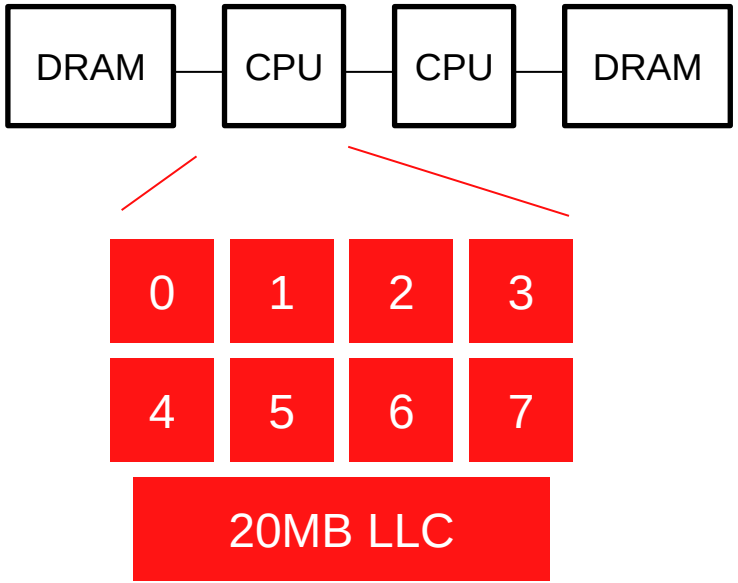
5. How to lay out data in memory – spread across memory banks for b/w, or keep close to a given CPU for latency?



How many schedulers do we need?

Analytics workload running on a shared cluster

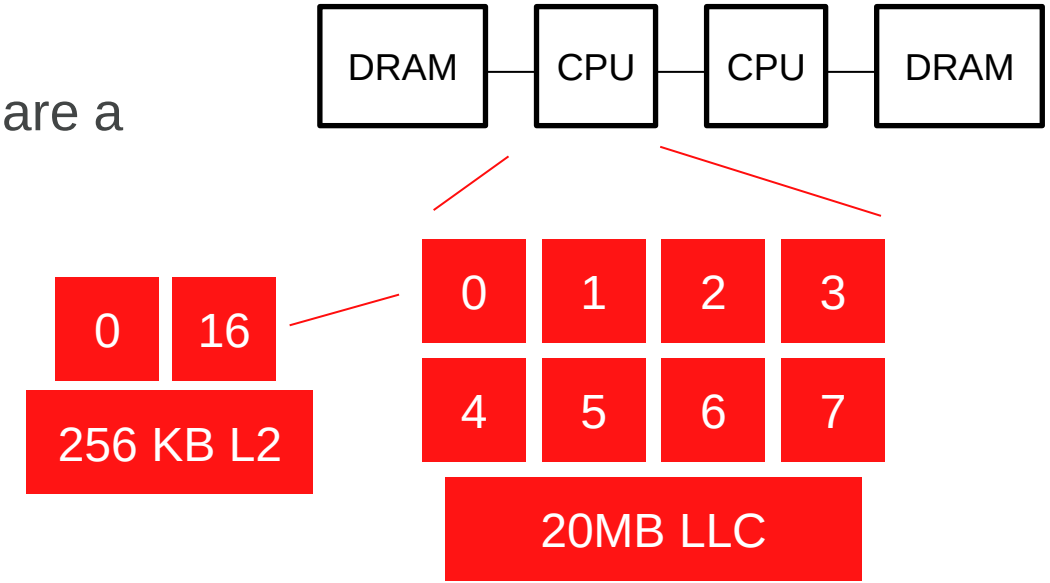
6. Where to place the computation?



How many schedulers do we need?

Analytics workload running on a shared cluster

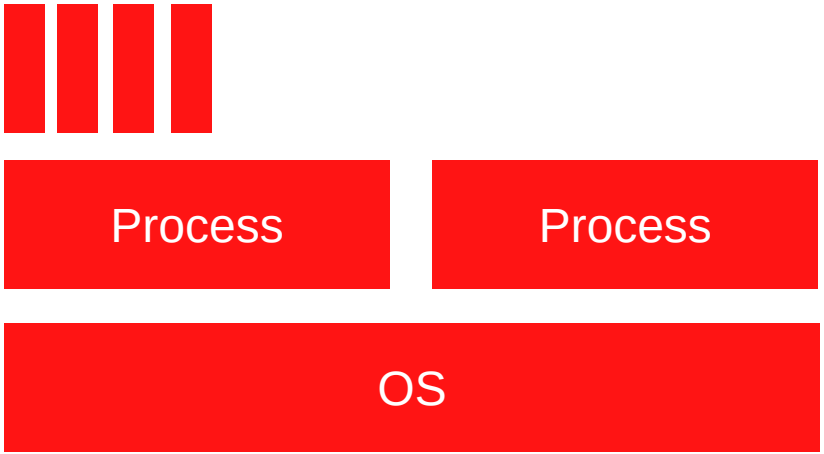
7. What to place on hyperthreads that share a core?



How many schedulers do we need?

Analytics workload running on a shared cluster

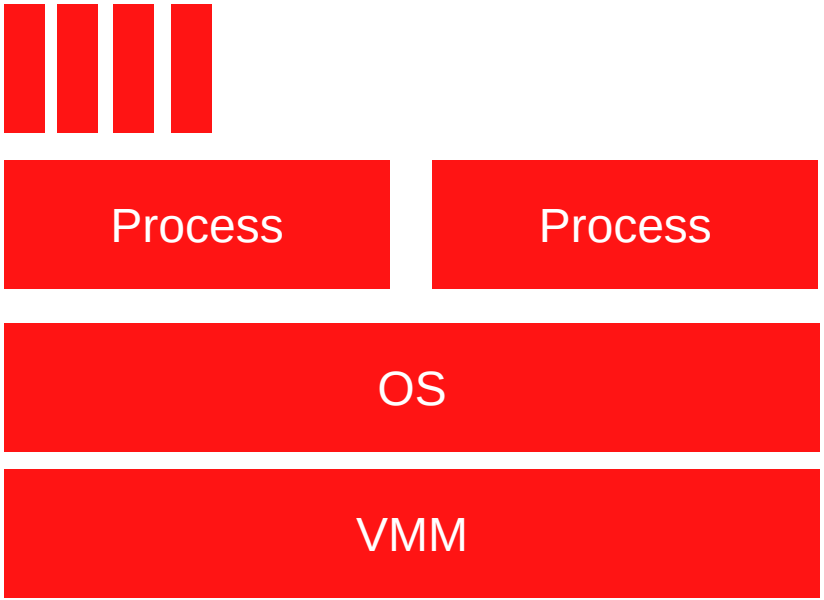
8. OS thread scheduler



How many schedulers do we need?

Analytics workload running on a shared cluster

9. VMM thread scheduler



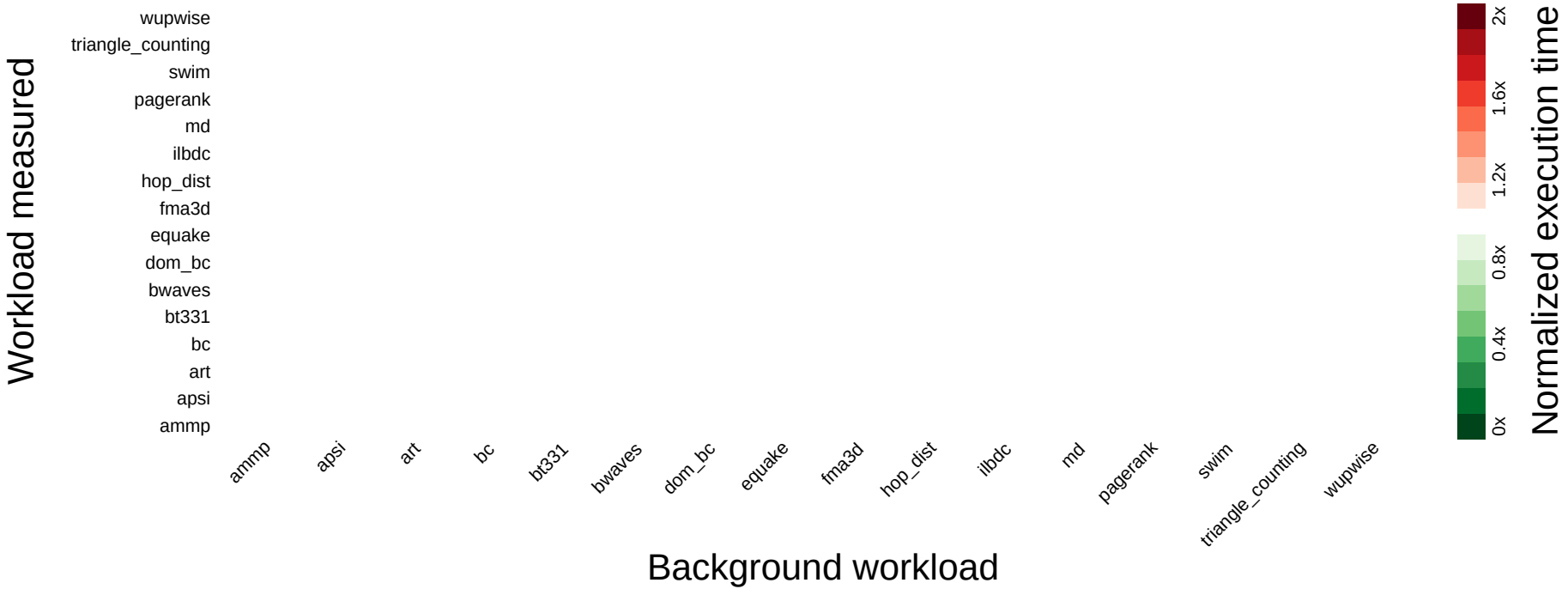
How many schedulers do we need?

Analytics workload running on a shared cluster

- And of course there are many more things going on as well
  - When to run GC or JIT, (lack of) co-ordination across distributed jobs
  - Background jobs, monitoring, profiling, etc.
  - Contention in the interconnect, on-chip or between machines
  - Live migration of VMs
  - Power management, thermal throttling
  - …

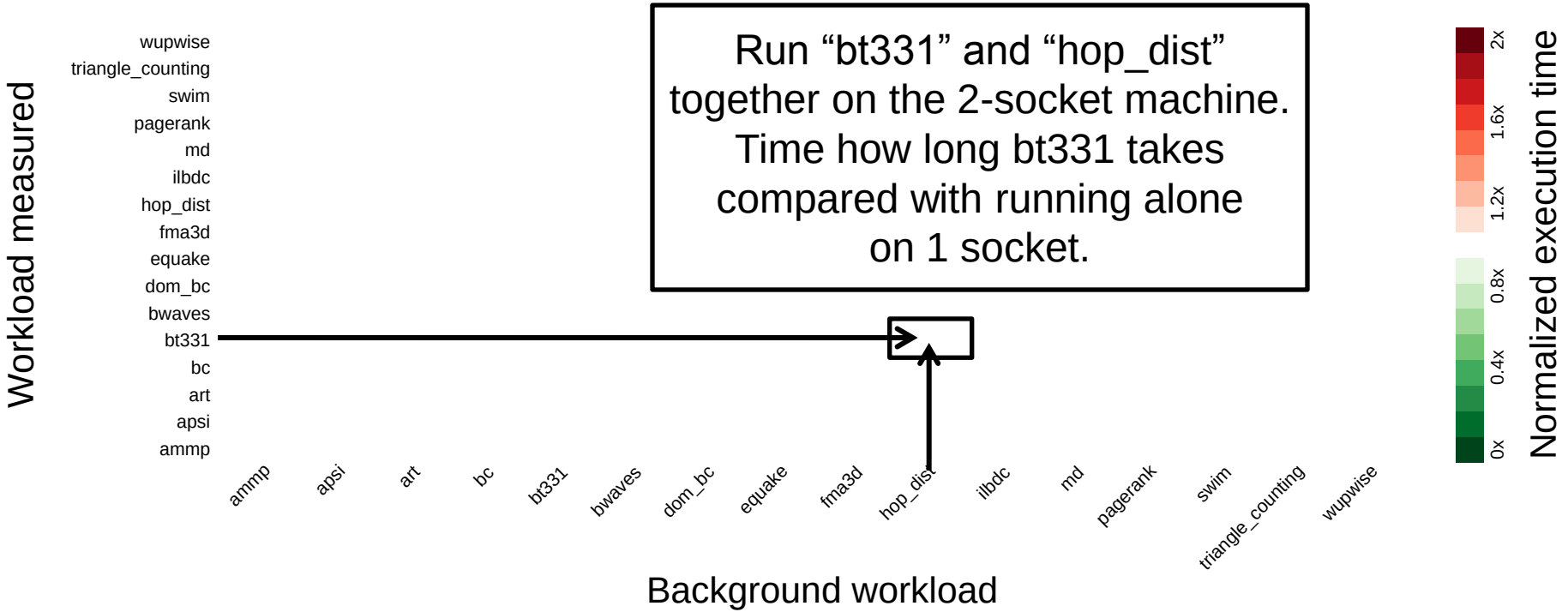
How bad is the problem?

2-socket Xeon E5-2660, Linux 2.6.32, GCC 4.8.0



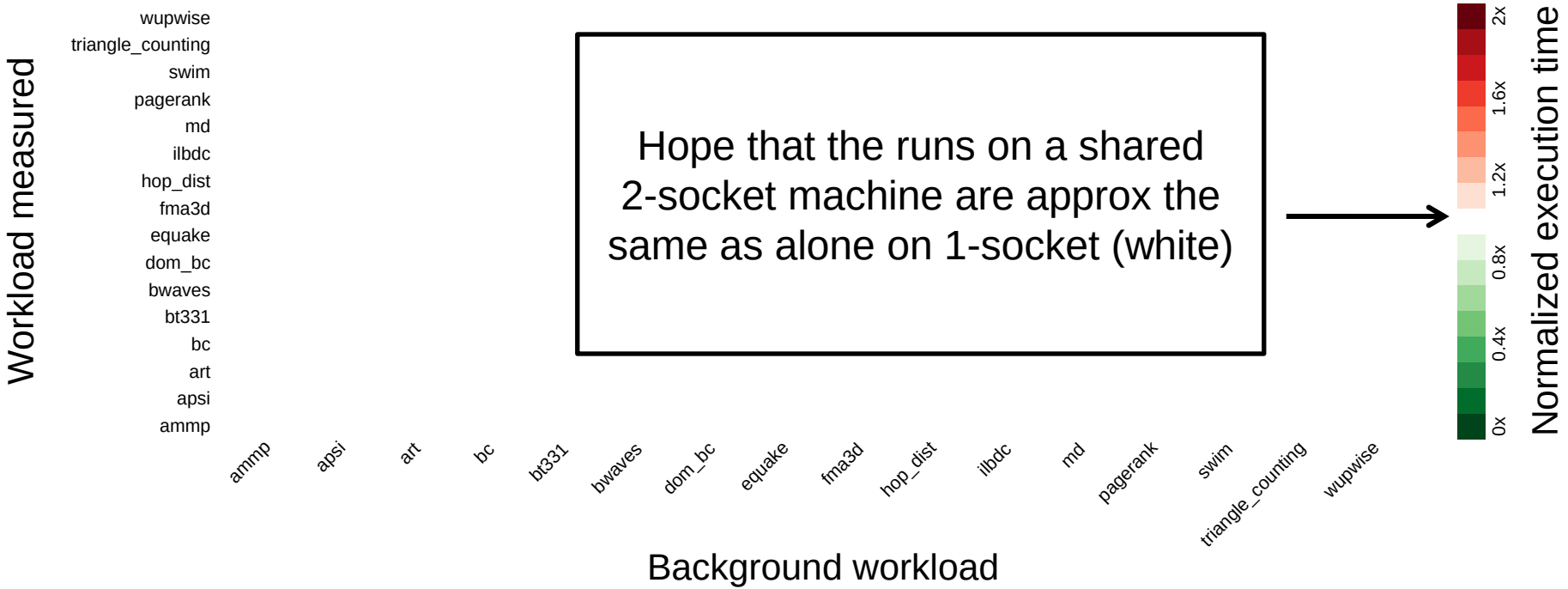
How bad is the problem?

2-socket Xeon E5-2660, Linux 2.6.32, GCC 4.8.0



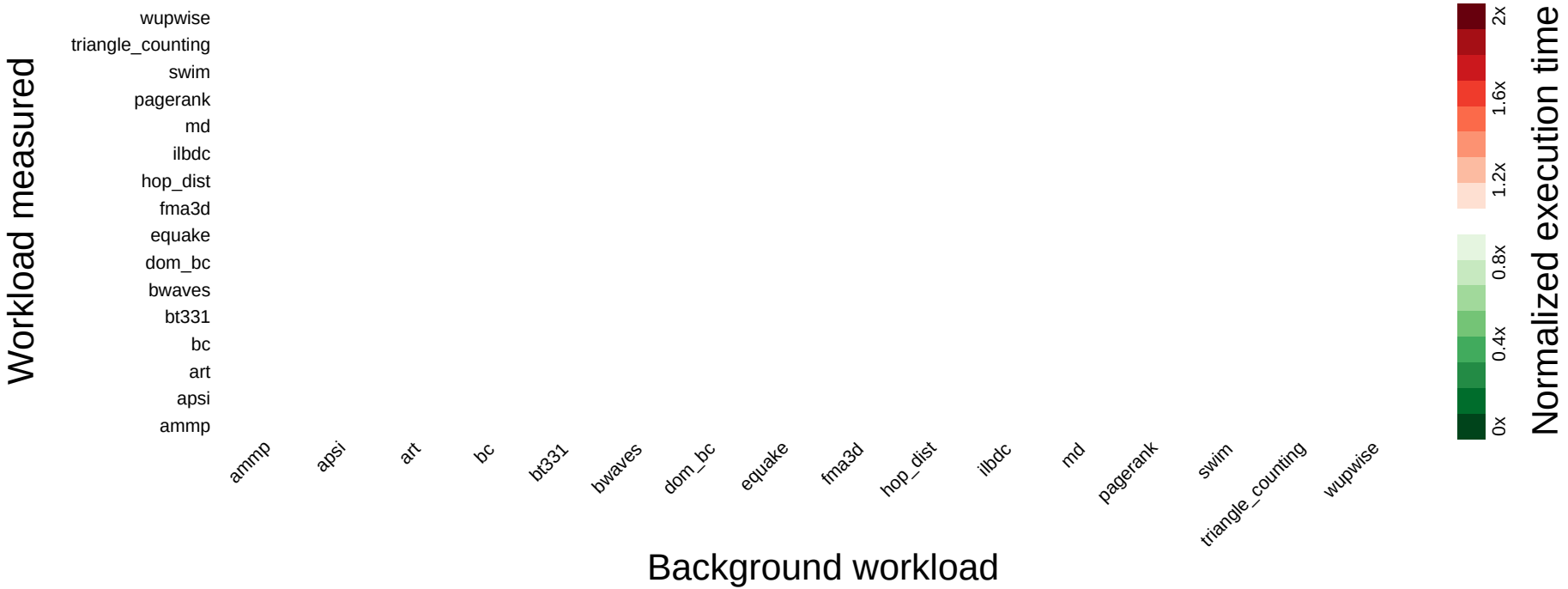
How bad is the problem?

2-socket Xeon E5-2660, Linux 2.6.32, GCC 4.8.0



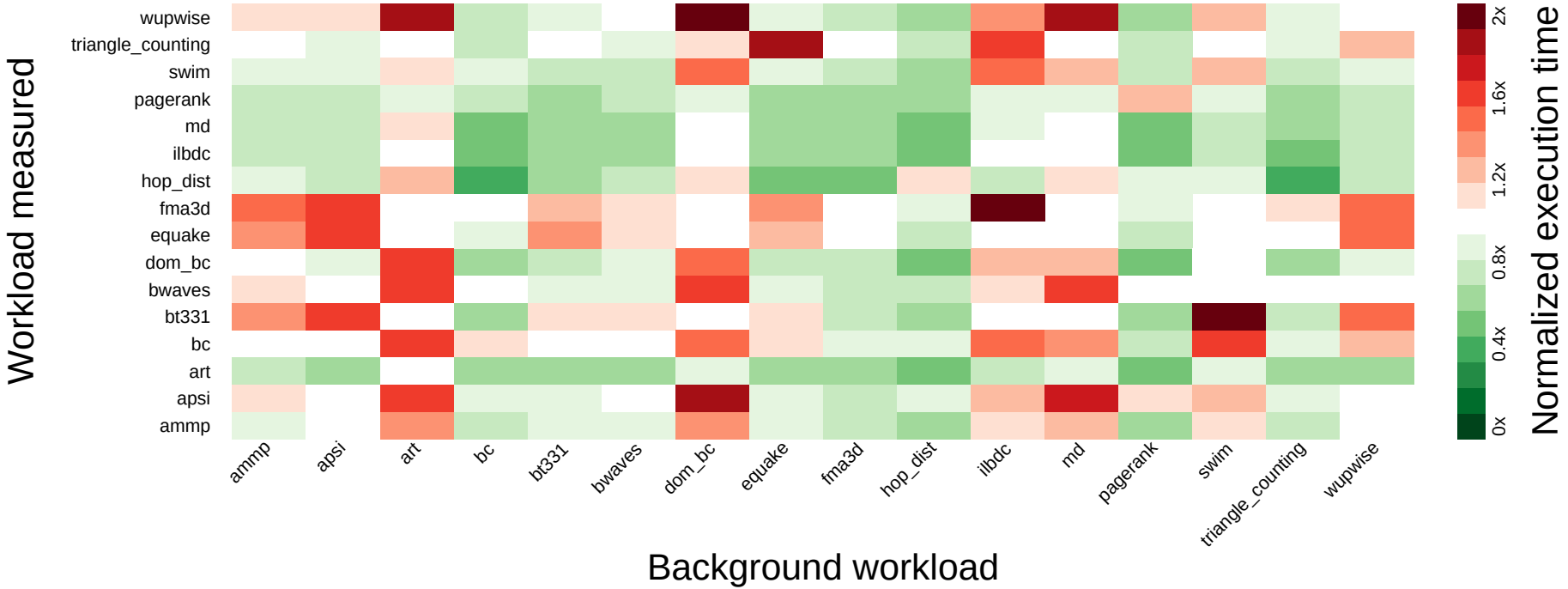
Instead of this…

2-socket Xeon E5-2660, Linux 2.6.32, GCC 4.8.0



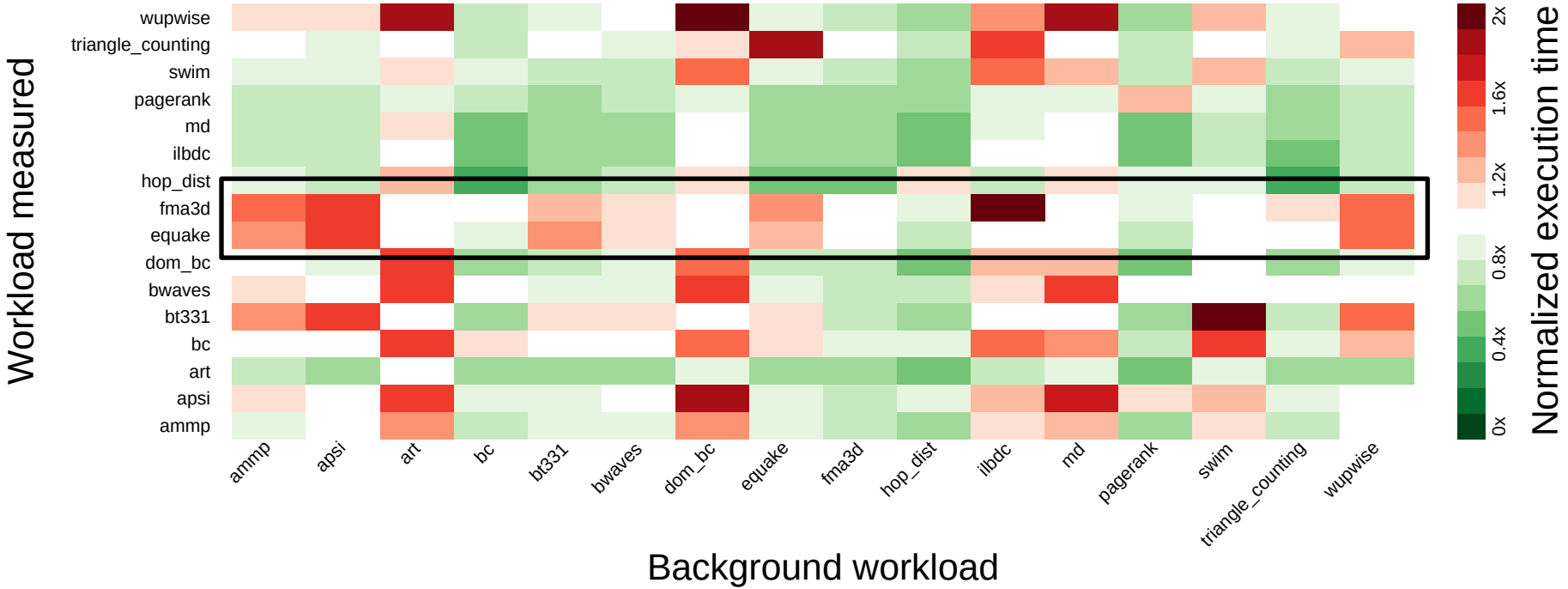
…it looks like this.

2-socket Xeon E5-2660, Linux 2.6.32, GCC 4.8.0



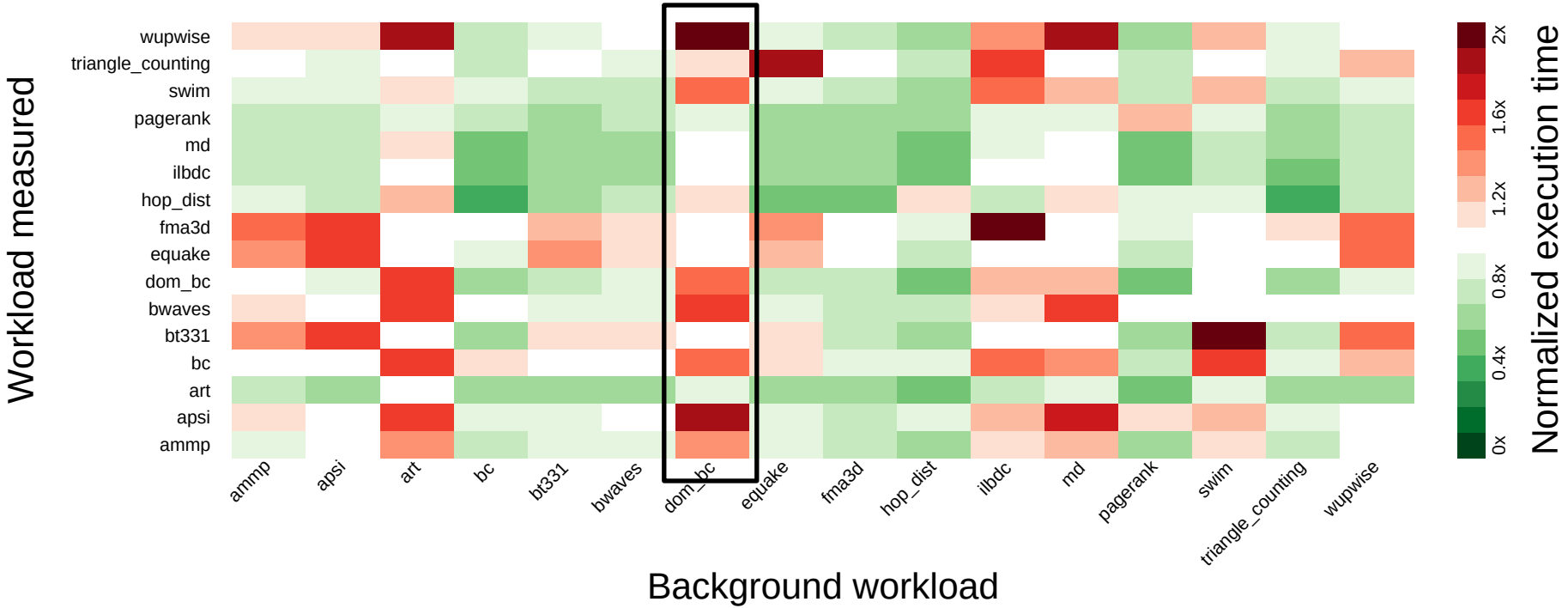
We see some patterns…

Red rows: sensitive applications



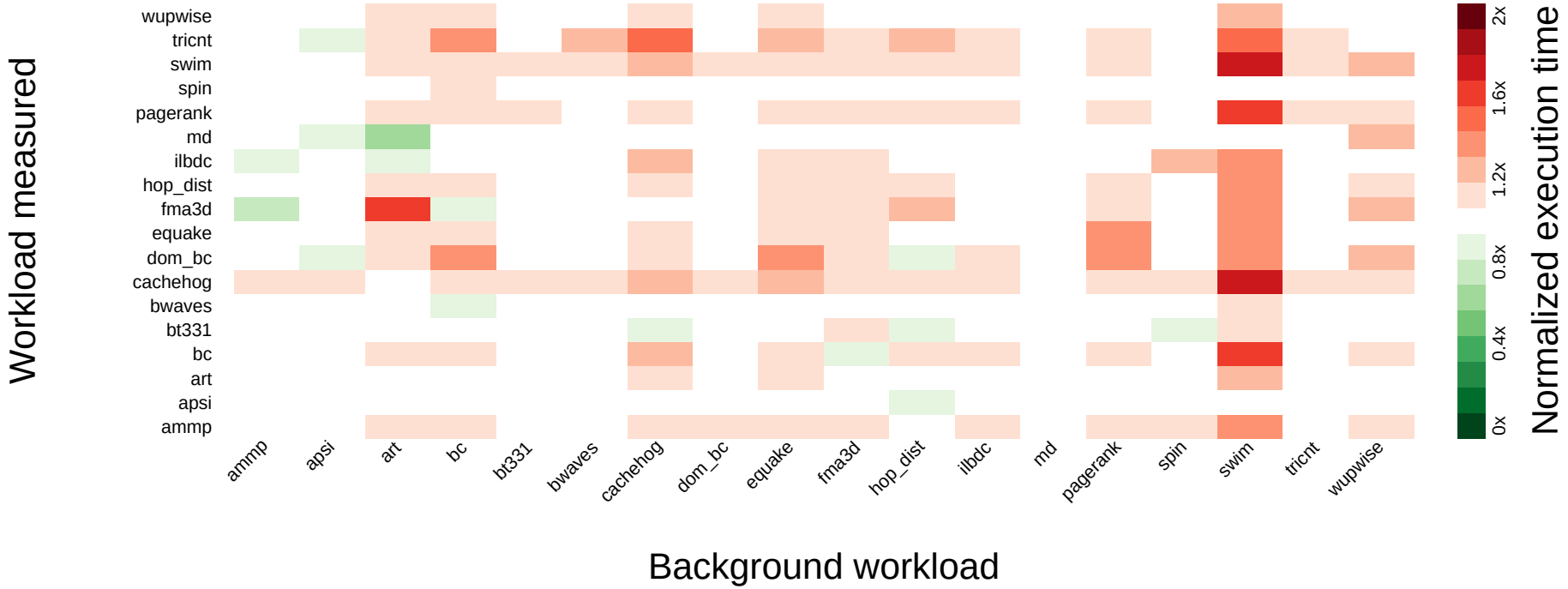
We see some patterns…

Red columns: aggressive applications



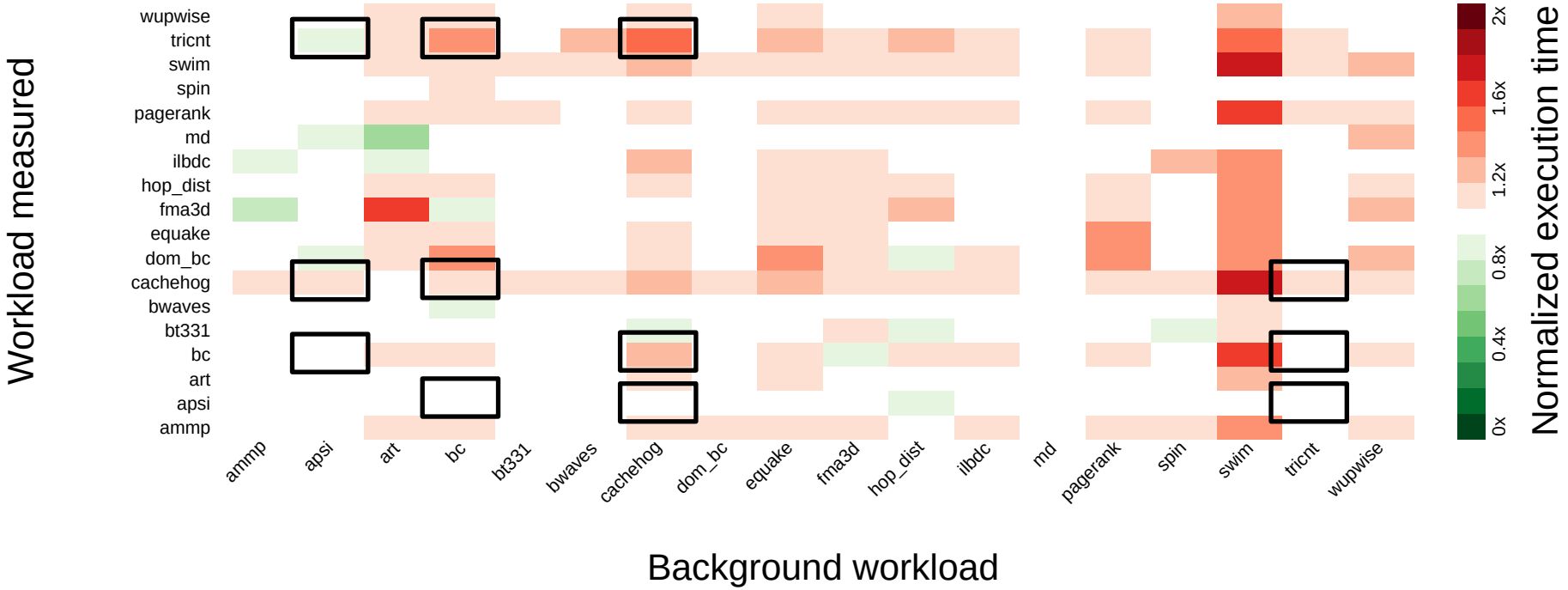
Sharing a socket

4-core jobs, sharing a socket vs separate sockets



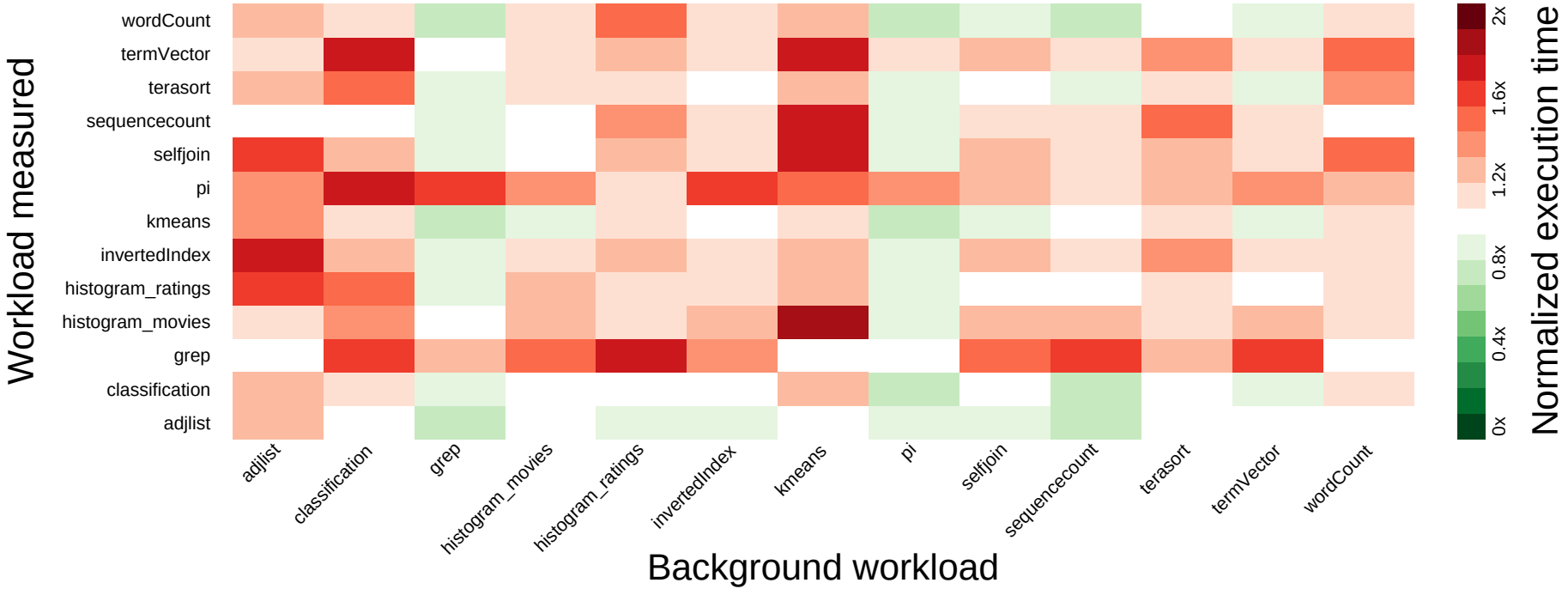
Sharing a socket

Given a set of jobs, which combinations are best?



Sharing a cluster

2 jobs on a shared cluster vs 1 job on half



Bad interactions at every level

(we've looked at so far)

- >2x difference between best and worst cases
- Often different results on different metrics
  - System-metrics (e.g., STP)
  - Per-job metrics (e.g., ANTT)

What can we do?

Over-provisioning

- No scheduling if there's no sharing
- Perhaps applicable to some resources
- But…
  - How to handle large bursty workloads?
  - The same problems remain within software (e.g., multiple parallel queries within an RDBMS)
  - Still need to use resources effectively when running on different machines, or within different allocations

What can we do?

Reduce performance fragility

- Trade some best-case performance for reduced sensitivity to dynamic scheduling decisions
- Lock-free algorithms
  - Don’t stall complete process if one thread is pre-empted or slow
- Over-partitioning
  - N processors, but divide work 4N ways and use work-stealing
- Cache-oblivious algorithms
  - Design algorithms to exploit a cache without knowing its size

What can we do?

Exploit knowledge of system structure

- Know the physical structure of the machine
- Know which core(s) and memory / interconnect is being used
- Tailor algorithms to exploit these resources
  - Pick from library of alternatives
  - Parameterize by resources
  - Use machine-learning to match algorithms and storage to systems?
- Make resource allocation/deallocation explicit to the application
  - Adapt when resources change

What can we do?

Vertically structures systems

- Eliminate or minimize sharing
  - "Layed Multiplexing Considered Harmful" – Tennenhouse '89
  - Nemesis research OS – Roscoe et al
- Provide abstraction via libraries, not servers
  - Push protection to h/w devices – now emerging via e.g. SR-IOV
- How would these ideas look in a rack-scale system?
  - E.g., considering communication between parts of a distributed job

What we talk about when we talk about scheduling

- Systems work from the early ’90s focused on resource management
  - With care, commodity hardware could handle audio and video
  - …but with time, over-provisioning removed the need for care
- Today, rack-scale systems can handle diverse workloads with substantial resource requirements
  - The problem now is sharing resources between jobs
  - Over-provisioning cannot be relied upon this time


ORACLE®