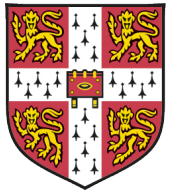


CamSaS

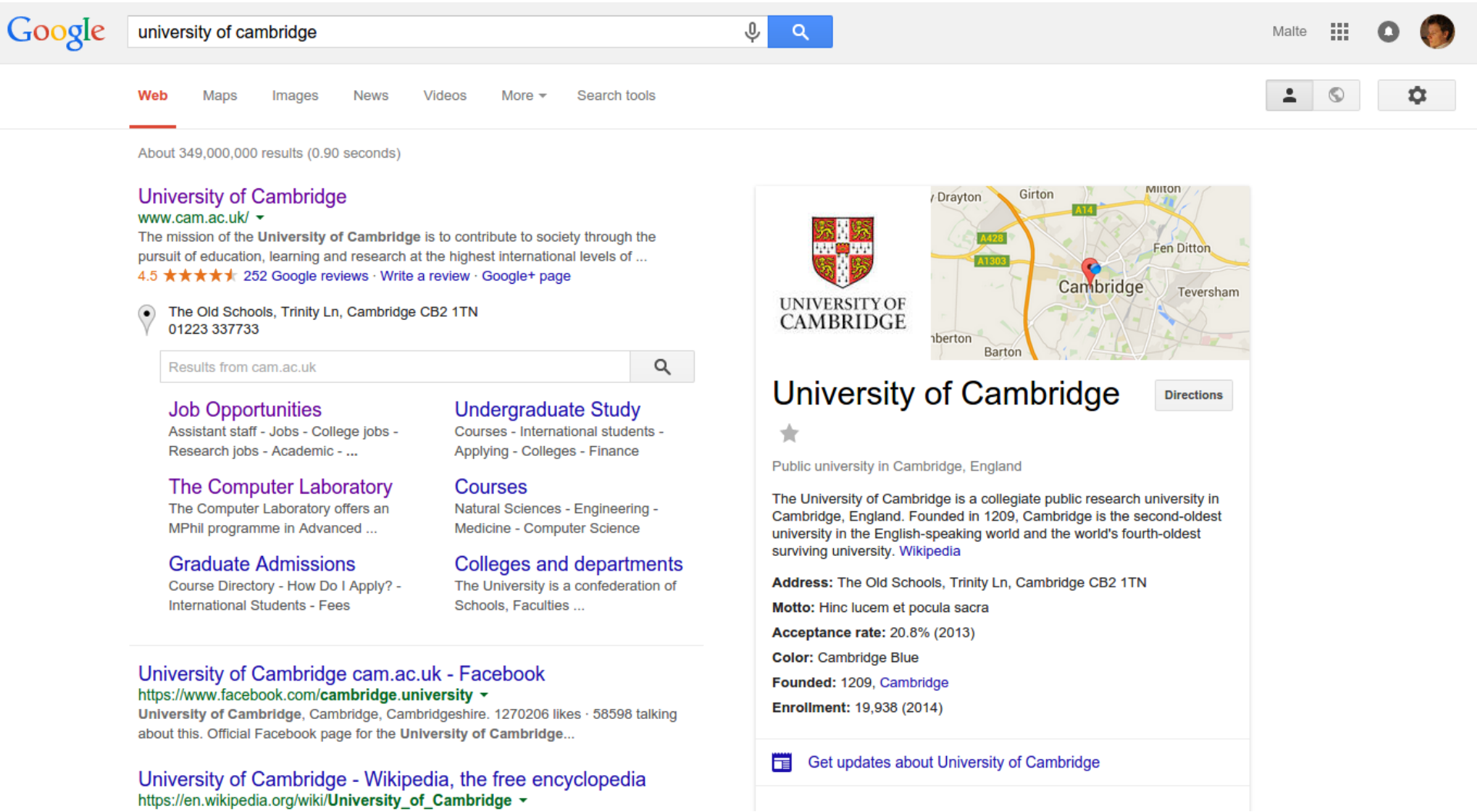
What does it take to make Google work at scale?

Malte Schwarzkopf

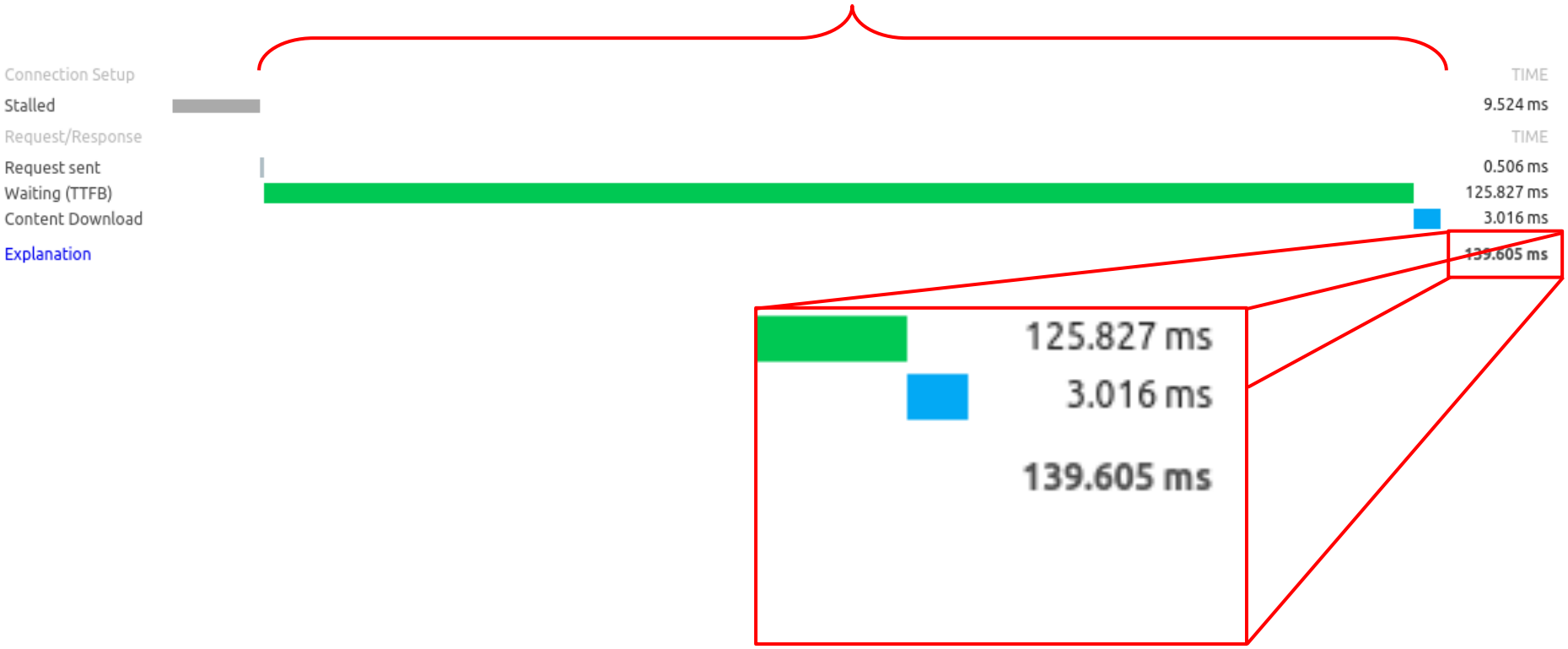
Cambridge Systems at Scale

http://camsas.org – @CamSysAtScale

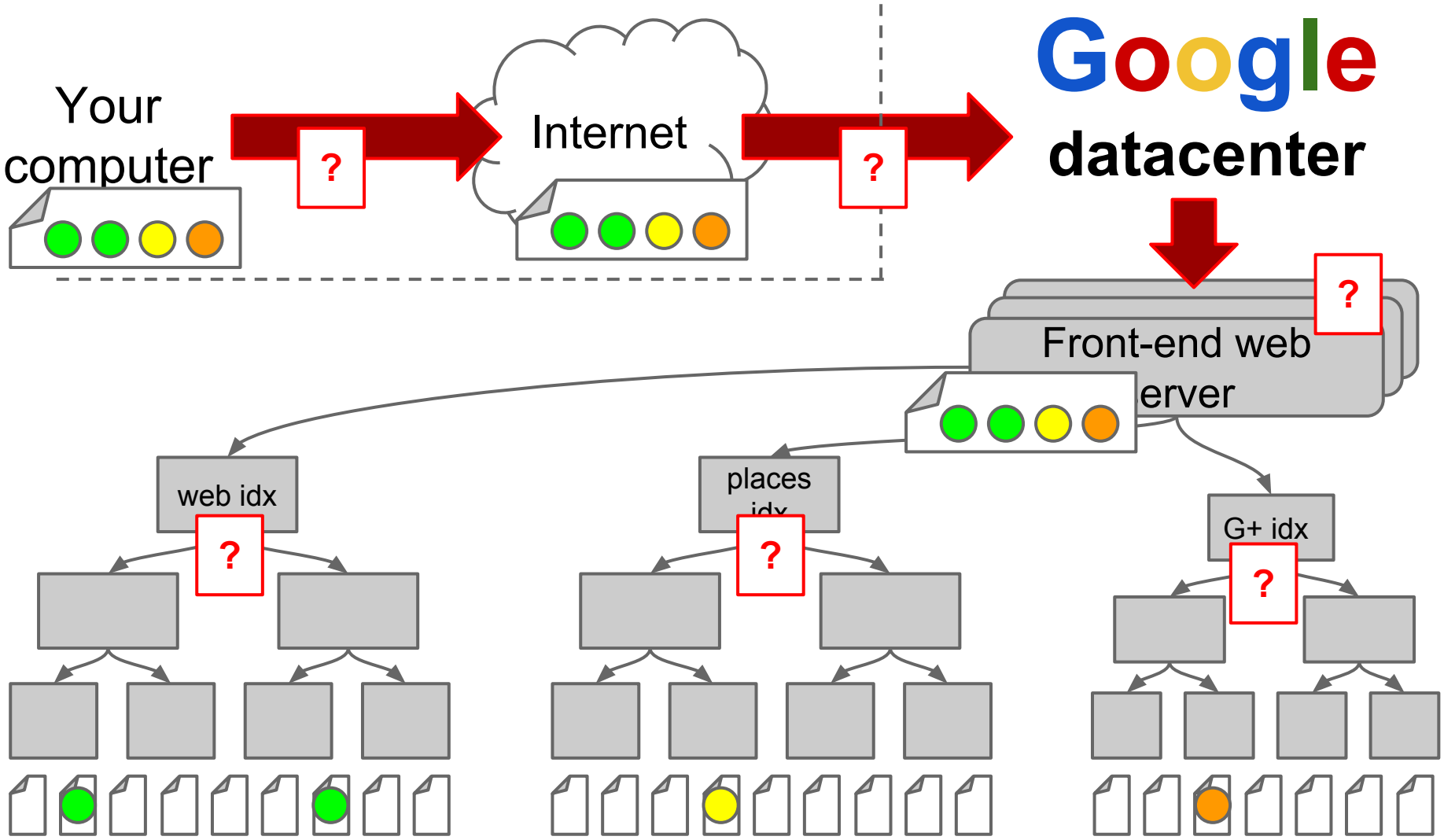

Google
UK
Google Search
I'm Feeling Lucky


Google
university of cambridge
Malte
Web
Maps
Images
News
Videos
More
Search tools
About 349,000,000 results (0.90 seconds)
University of Cambridge
www.cam.ac.uk/
The mission of the University of Cambridge is to contribute to society through the pursuit of education, learning and research at the highest international levels of ...
4.5 252 Google reviews · Write a review · Google+ page
The Old Schools, Trinity Ln, Cambridge CB2 1TN
01223 337733
Results from cam.ac.uk
Job Opportunities
Assistant staff - Jobs - College jobs - Research jobs - Academic - ...
Undergraduate Study
Courses - International students - Applying - Colleges - Finance
The Computer Laboratory
The Computer Laboratory offers an MPhil programme in Advanced ...
Courses
Natural Sciences - Engineering - Medicine - Computer Science
Graduate Admissions
Course Directory - How Do I Apply? - International Students - Fees
Colleges and departments
The University is a confederation of Schools, Faculties ...
University of Cambridge cam.ac.uk - Facebook
https://www.facebook.com/cambridge.university
University of Cambridge, Cambridge, Cambridgeshire. 1270206 likes · 58598 talking about this. Official Facebook page for the University of Cambridge...
University of Cambridge - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/University_of_Cambridge
UNIVERSITY OF CAMBRIDGE
Drayton
Girton
A14
A428
A1303
Fen Ditton
Cambridge
Teversham
Barton
University of Cambridge
Directions
Public university in Cambridge, England
The University of Cambridge is a collegiate public research university in Cambridge, England. Founded in 1209, Cambridge is the second-oldest university in the English-speaking world and the world's fourth-oldest surviving university. Wikipedia
Address: The Old Schools, Trinity Ln, Cambridge CB2 1TN
Motto: Hinc lucem et pocula sacra
Acceptance rate: 20.8% (2013)
Color: Cambridge Blue
Founded: 1209, Cambridge
Enrollment: 19,938 (2014)
Get updates about University of Cambridge

What happens here?



What happens in those 139ms?

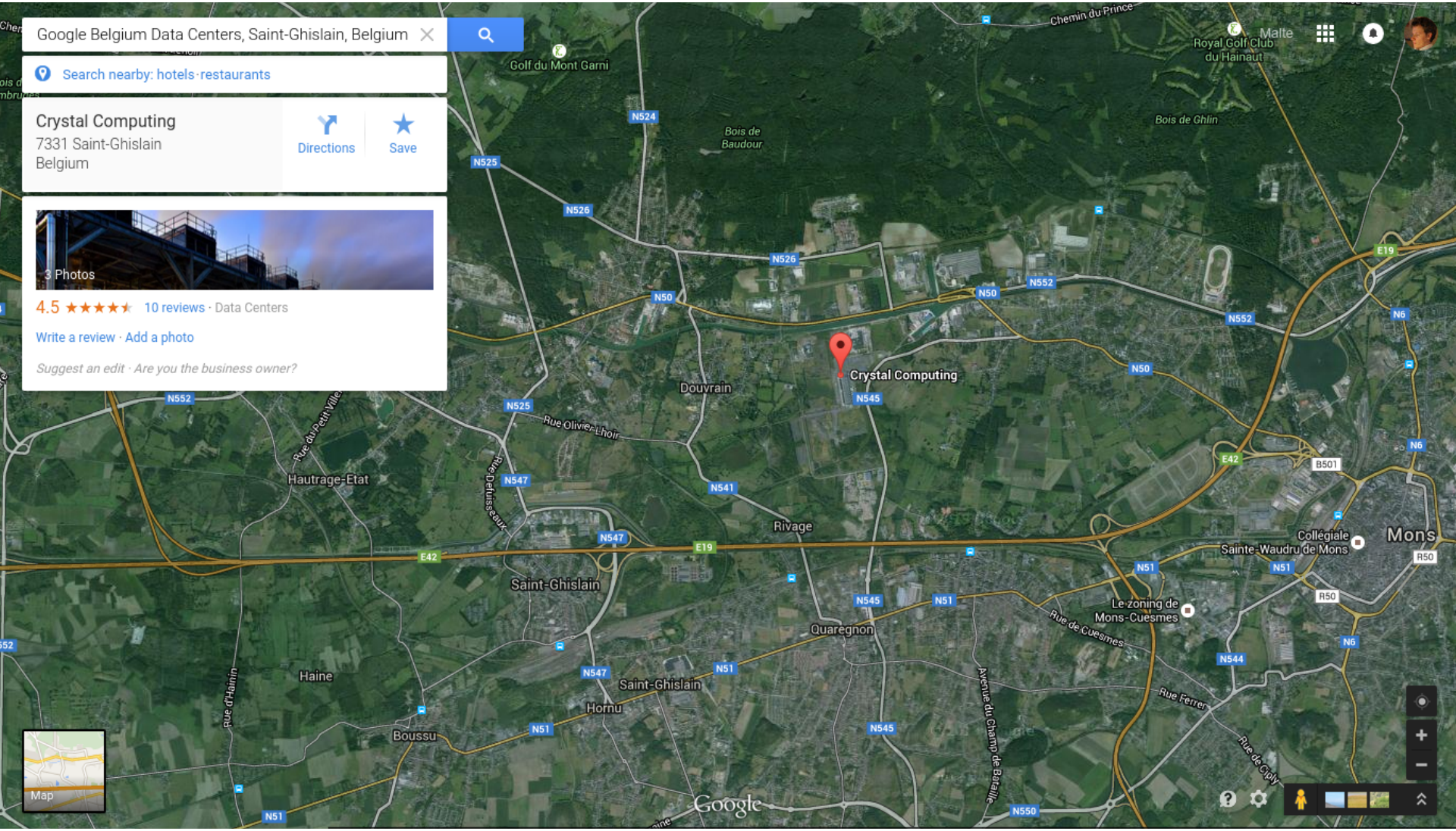


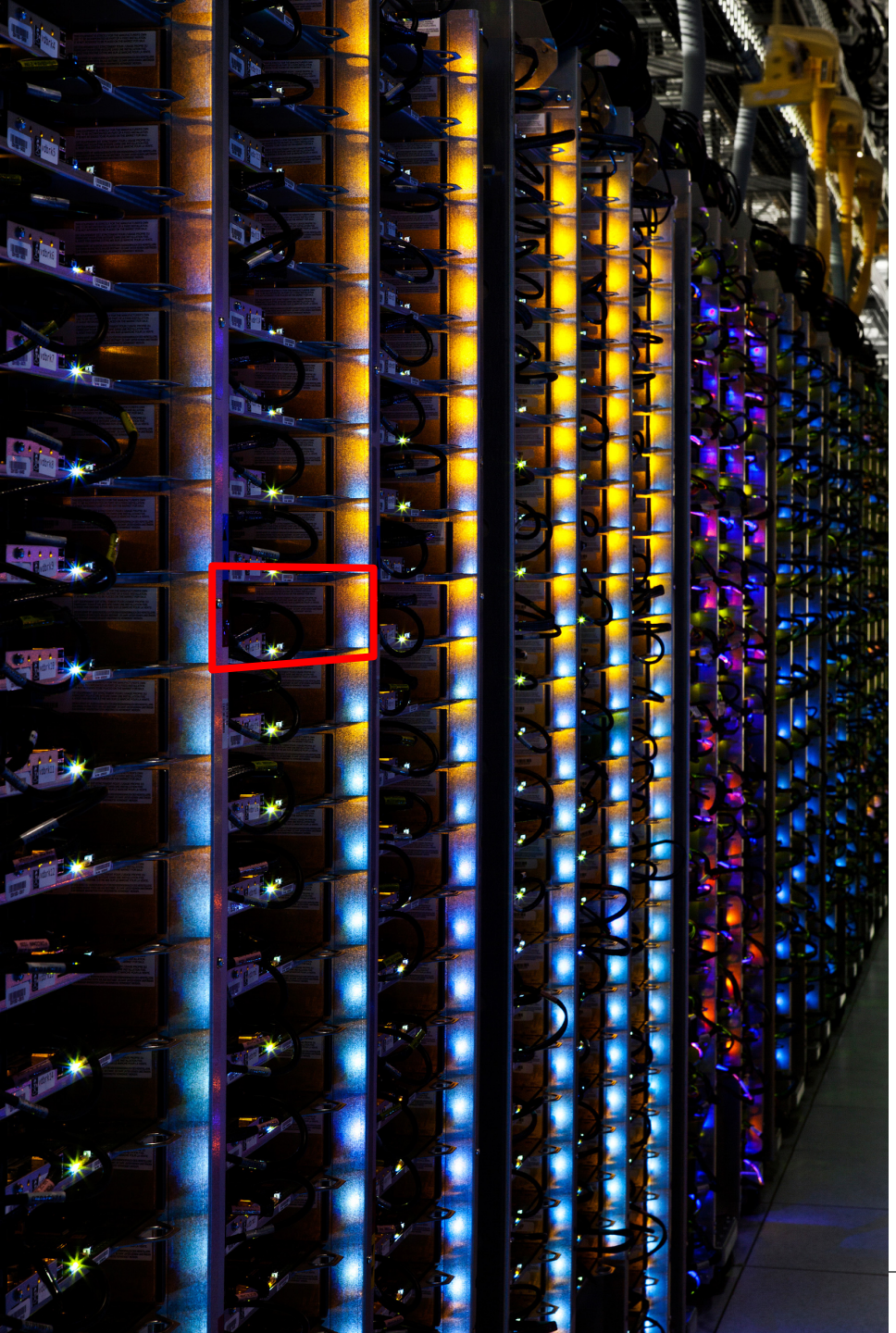
What we'll chat about

1. Datacenter hardware
2. Scalable software stacks
    a. Google
    b. Facebook
3. Differences compared to HPC
4. Current research directions

Please ask questions - any time! :)


Liverpool
Chester
Sheffield
Peak District National Park
Snowdonia
Nottingham
Leicester
Peterborough
Norwich
ENGLAND
WALES
Ipswich
The Cotswolds AONB
Oxford
Swansea
Cardiff
Bristol
Swindon
Reading
London
Bath
North Wessex Downs AONB
Canterbury
Southampton
Brighton
Exeter
Bournemouth
Portsmouth
Torquay
Plymouth
St Austell
English Channel
Guernsey
Le Havre
Rouen
Caen
Amiens
Lens
Calais
Bruges
Ghent
Antwerp
Brussels
Lille
Crystal Computing
Alkmaar
Amsterdam
The Hague
Netherlands
Arnhem
Rotterdam
Eindhoven
Groningen
Leer
Oldenburg
Münster
Dortmund
Essen
Düsseldorf
Cologne
Bonn
Aachen
Liège
Koblenz
Wiesbaden
Luxembourg City
Trier
Reims


Google Belgium Data Centers, Saint-Ghislain, Belgium
Search nearby: hotels·restaurants
Crystal Computing
7331 Saint-Ghislain
Belgium
Directions
Save
3 Photos
4.5 ★★★★½ 10 reviews · Data Centers
Write a review · Add a photo
Suggest an edit · Are you the business owner?
Crystal Computing
Douvrain
Rivage
Hautrage-Etat
Saint-Ghislain
Quaregnon
Haine
Hornu
Boussu
Mons
Collégiale Sainte-Waudru de Mons
Le zoning de Mons-Cuesmes
Golf du Mont Garni
Royal Golf Club du Hainaut
Malte
Bois de Baudour
Bois de Ghlin
Chemin du Prince
Rue Olivier-Lhoir
Rue du Petit Ville
Rue de Cuesmes
Rue Ferrer
Rue de Ciply
Avenue du Champ de Bataille
Rue d'Hainin
N524
N525
N526
N50
N552
N545
N547
N541
N51
N544
N550
E19
E42
B501
N6
R50
Map
Google

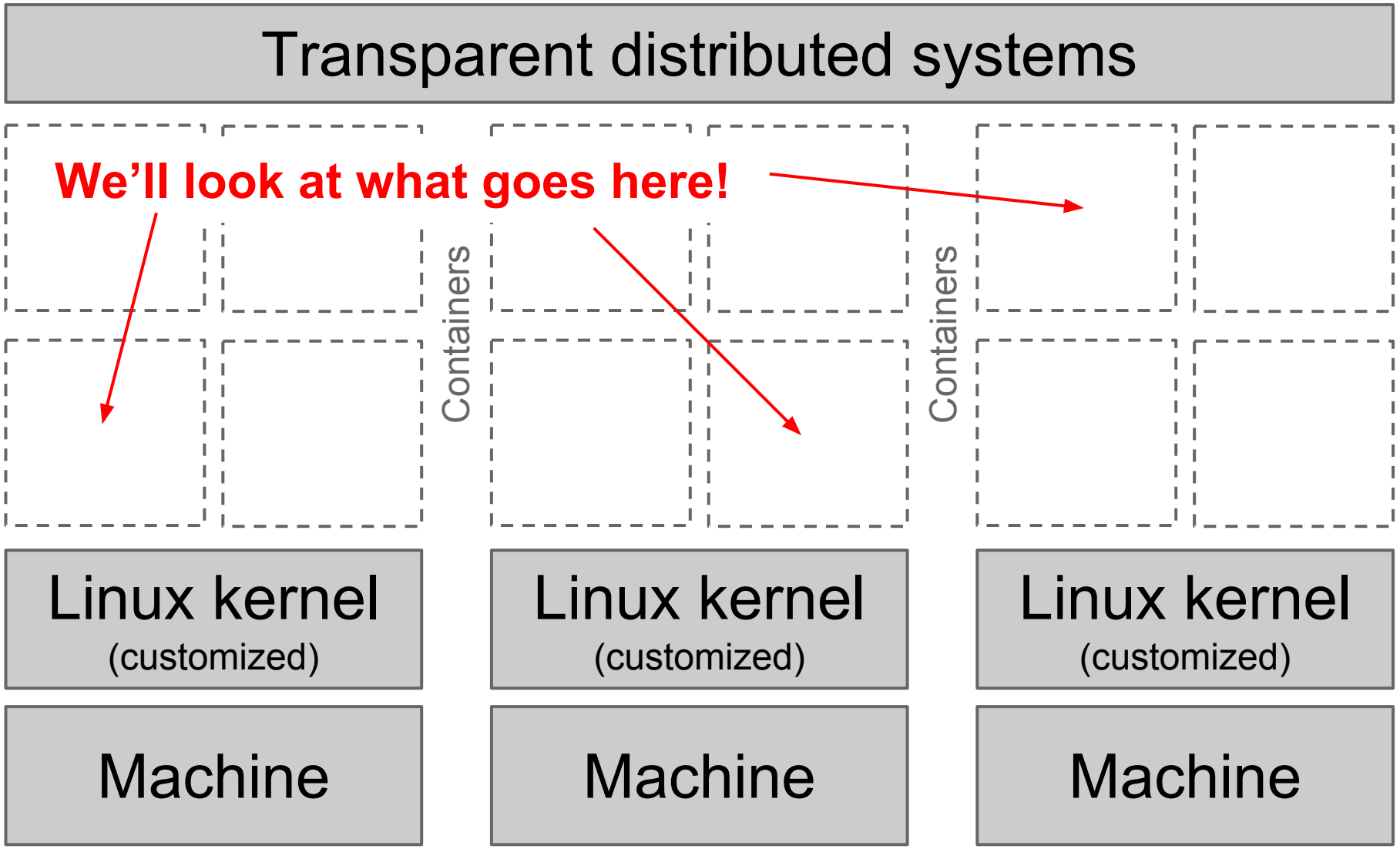


- 10,000+ machines
- “Machine”
  - no chassis!
  - DC battery
  - ~6 disk drives
  - mostly custom-made
- Network
  - ToR switch
  - multi-path core
  - e.g., 3-stage fat tree or Clos topology

How's this different to HPC?

- Emphasis on **commodity hardware**
  - **No expensive interconnect**
  - Mid-range machines
  - Energy/performance/cost trade-off essential
- Massive **automation**
  - Very small number of on-site staff
  - Automated software bootstrap
- **Fault tolerant** design
  - Each component can fail
  - Software must be aware and compensate

The software side



The Google Stack

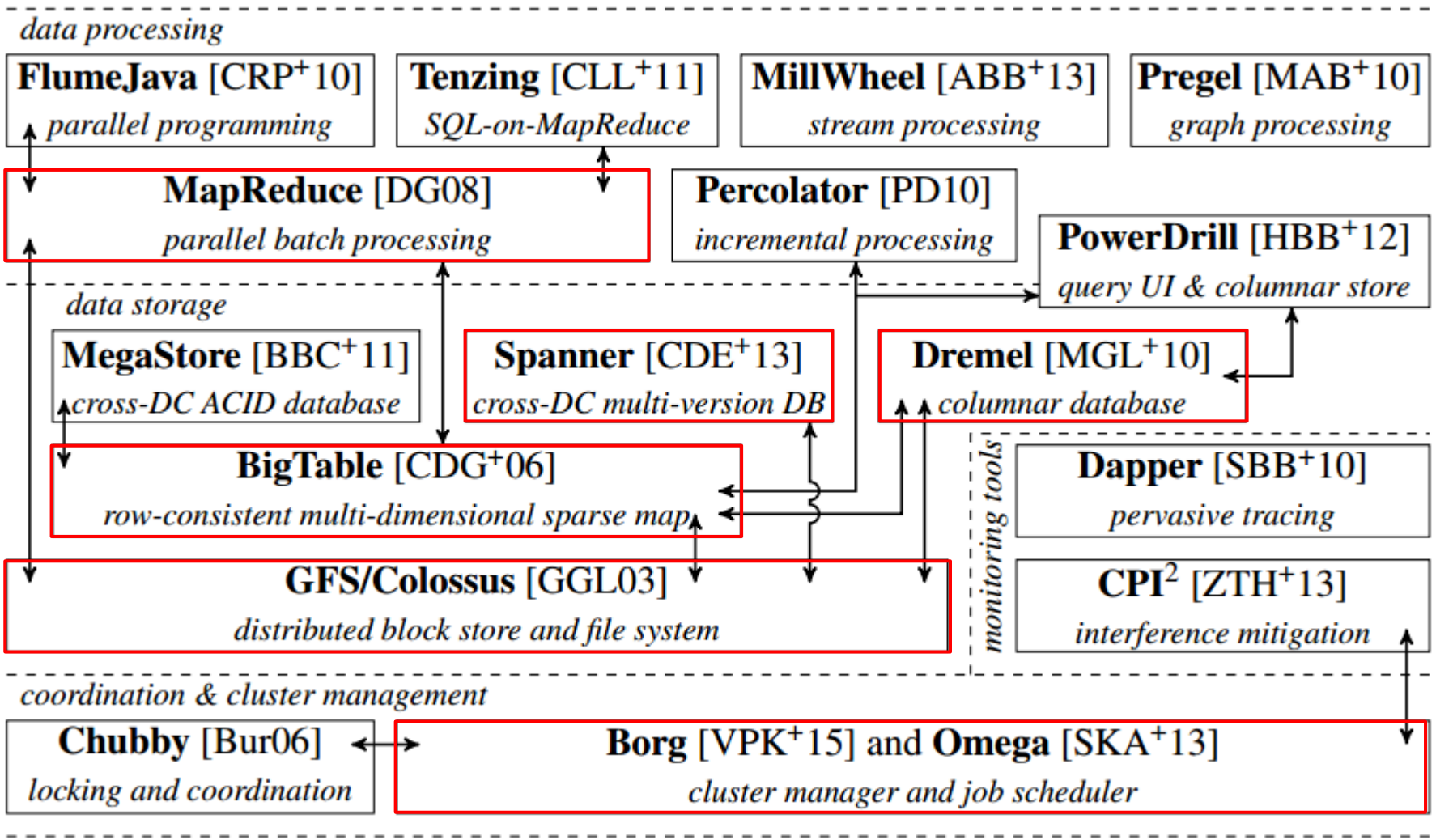


Figure from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

The Google Stack

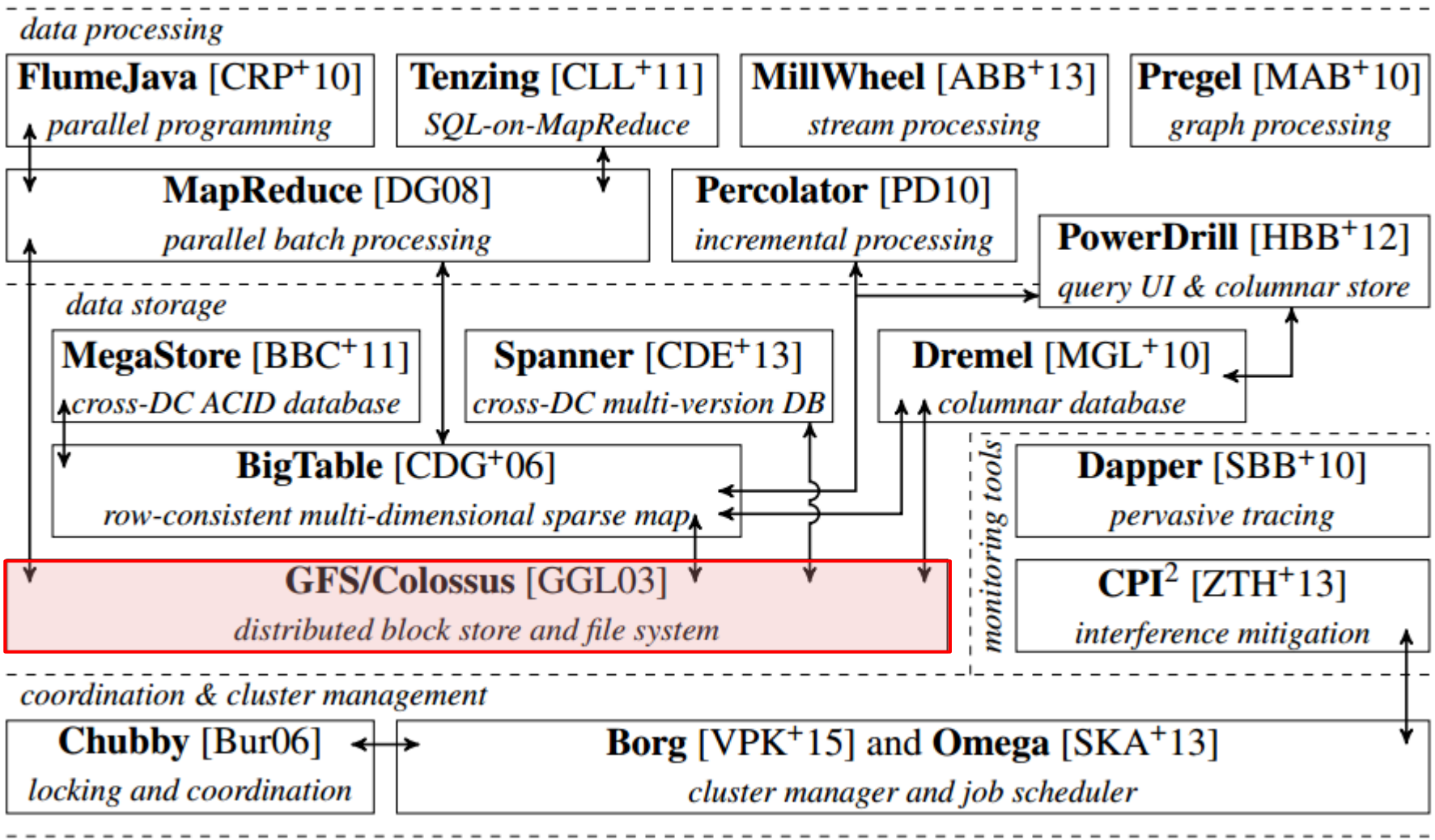


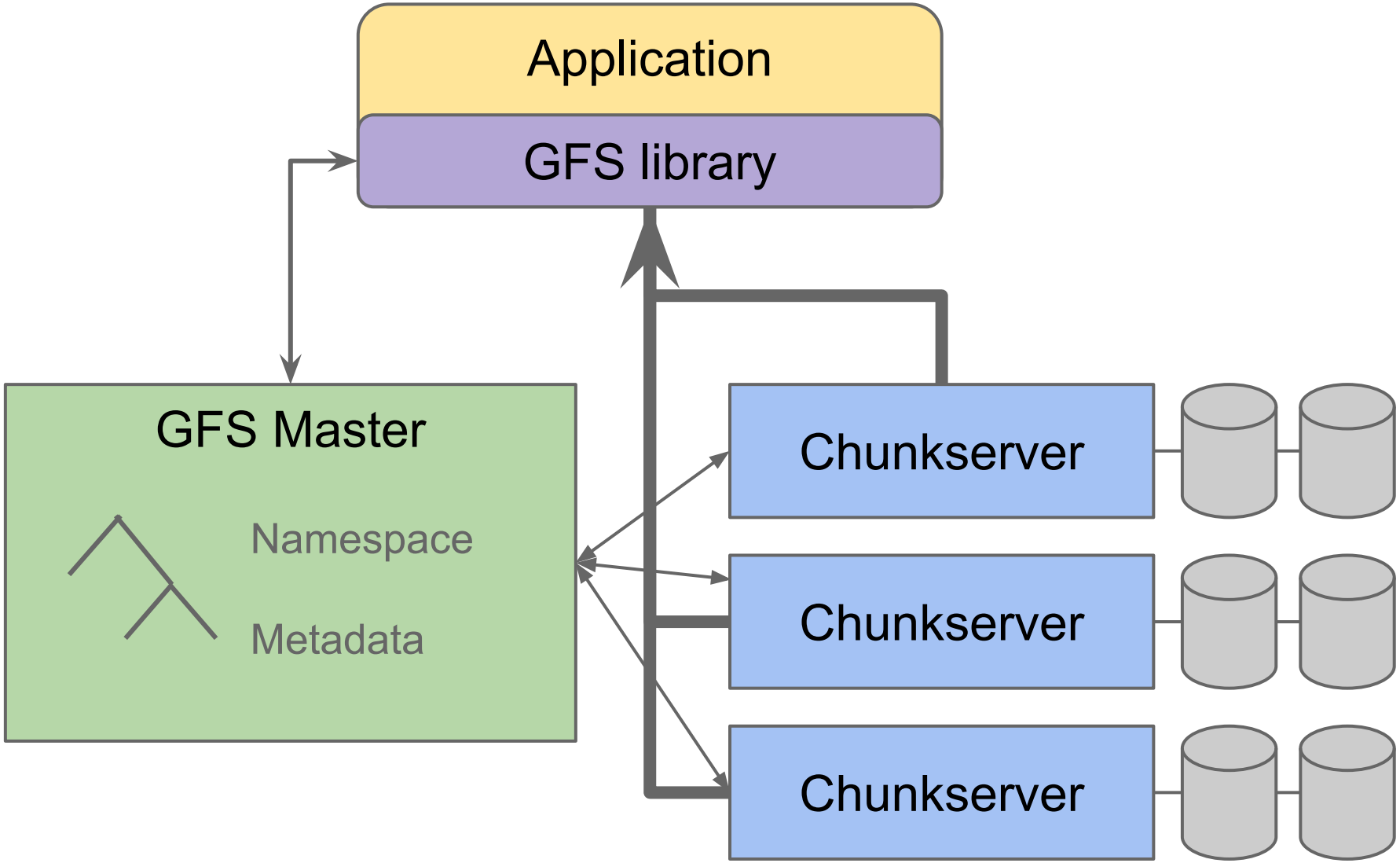
Figure from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

GFS/Colossus

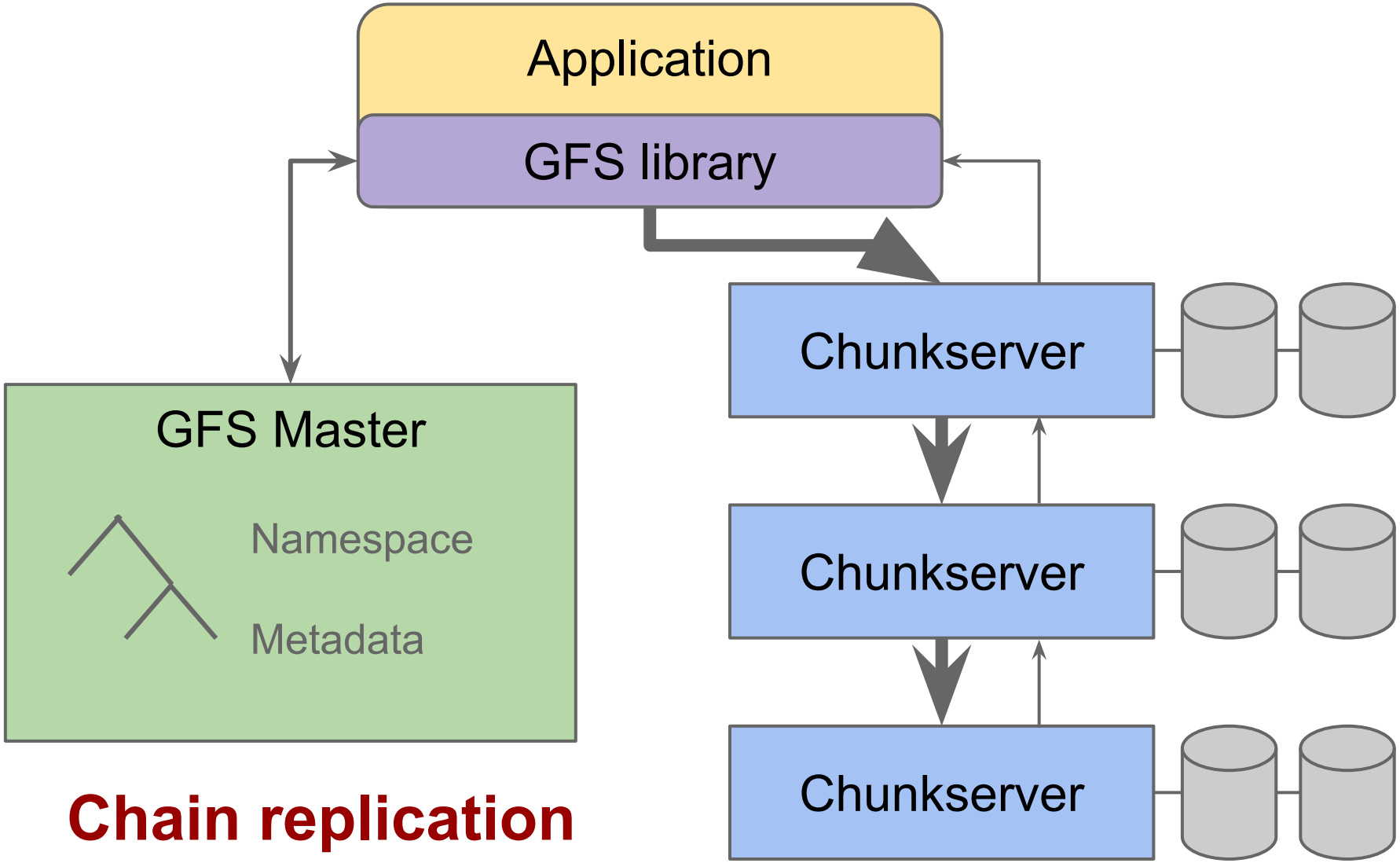
- Bulk data block storage system
  - Optimized for large files (GB-size)
  - Supports small files, but not common case
  - **Read, write, append** modes

- **Colossus** = GFSv2, adds some improvements
  - e.g., Reed-Solomon-based erasure coding
  - better support for latency-sensitive applications
  - **sharded meta-data** layer, rather than single master

GFS/Colossus: architecture

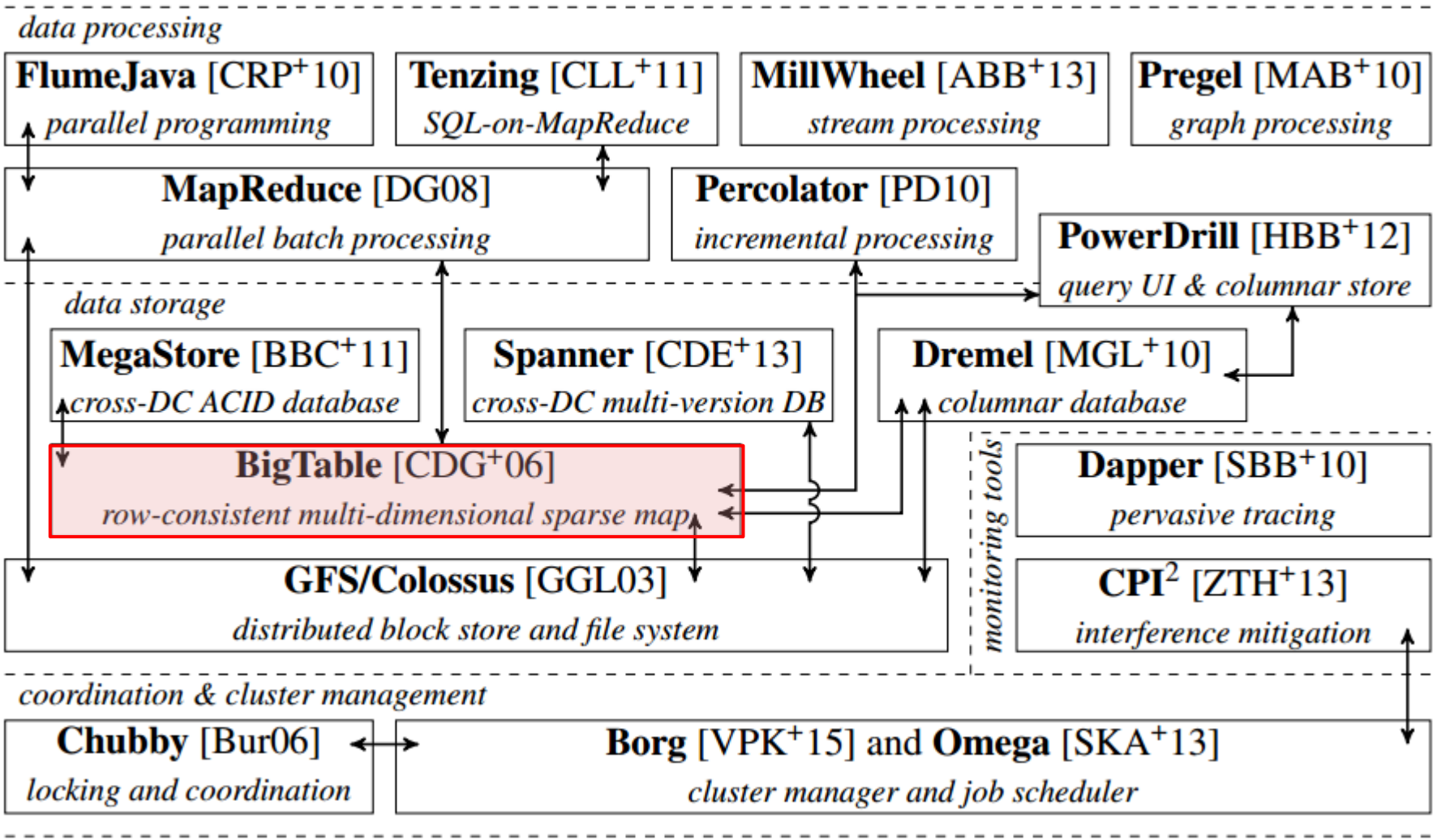


GFS/Colossus: writes



Chain replication

The Google Stack

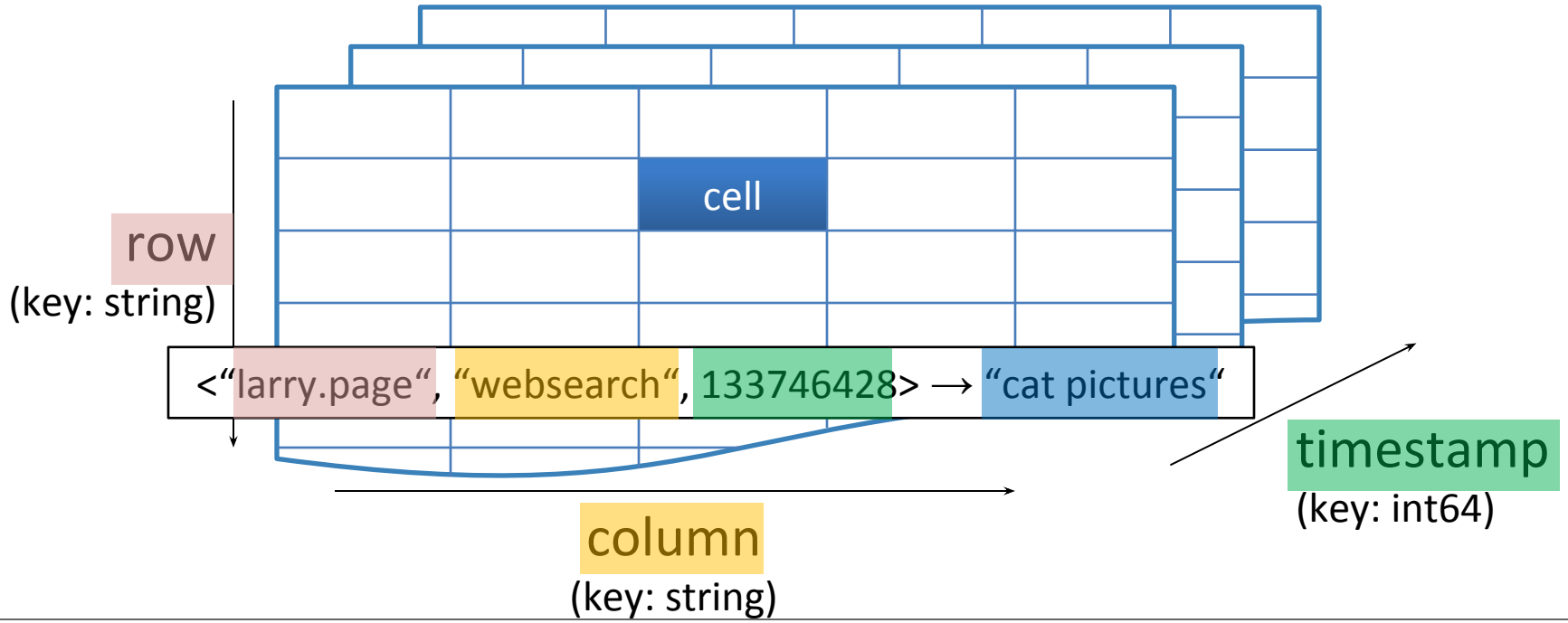


Details & Bibliography: http://malteschwarzkopf.de/research/assets/google-stack.pdf

Figure from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

BigTable (2006)

- ‘Three-dimensional‘ key-value store:
  - *<row key, column key, timestamp> → value*
- Effectively a distributed, sorted, sparse map



BigTable (2006)

- Distributed **tablets** (~1 GB) hold subsets of map
- On top of Colossus, which handles replication and fault tolerance: *only one (active) server per tablet!*
- Reads & writes within a row are **transactional**
  - Independently of the number of columns touched
  - **But:** no cross-row transactions possible
  - Turns out users find this hard to deal with

BigTable (2006)

- META0 tablet is “root“ for name resolution
  - Like a coordinator/leader
  - Nested META1 tablets improve meta-data lookup
- Elect master (META0 tablet server) using **Chubby**, and to maintain list of tablet servers & schemas
  - 5-way replicated Paxos consensus on data in Chubby
- Built atop GFS; building block for MegaStore
  - “Next gen” with transactions: Spanner

The Google Stack

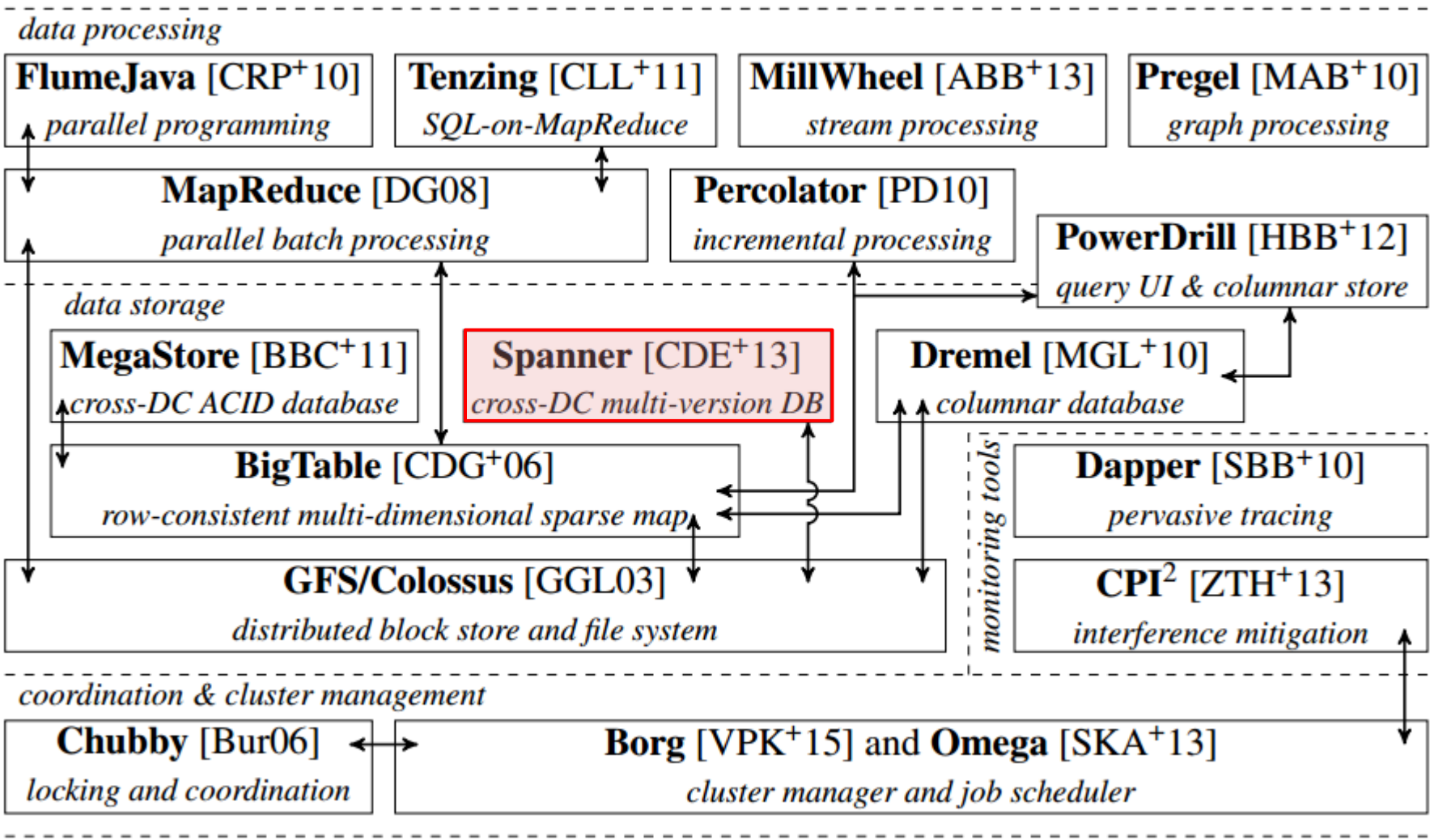


Figure from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

Spanner (2012)

- BigTable insufficient for some consistency needs
- Often have transactions across >1 data centres
  - May buy app on Play Store while travelling in the U.S.
  - Hit U.S. server, but customer billing data is in U.K.
  - Or may need to update several replicas for fault tolerance

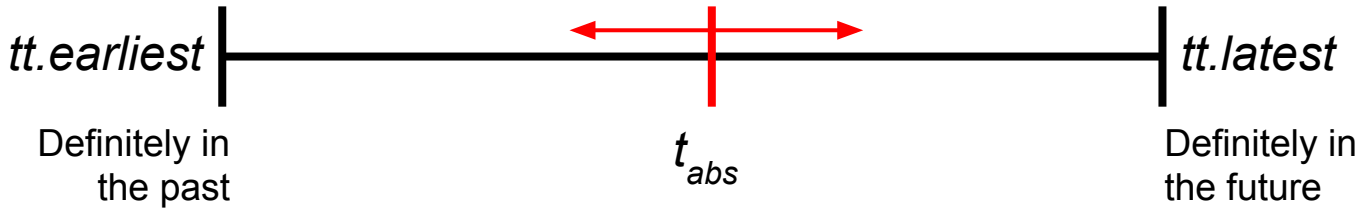
- Wide-area consistency is hard
  - due to long delays and clock skew
  - no **global, universal notion of time**
  - NTP not accurate enough, PTP doesn't work (jittery links)

Spanner (2012)

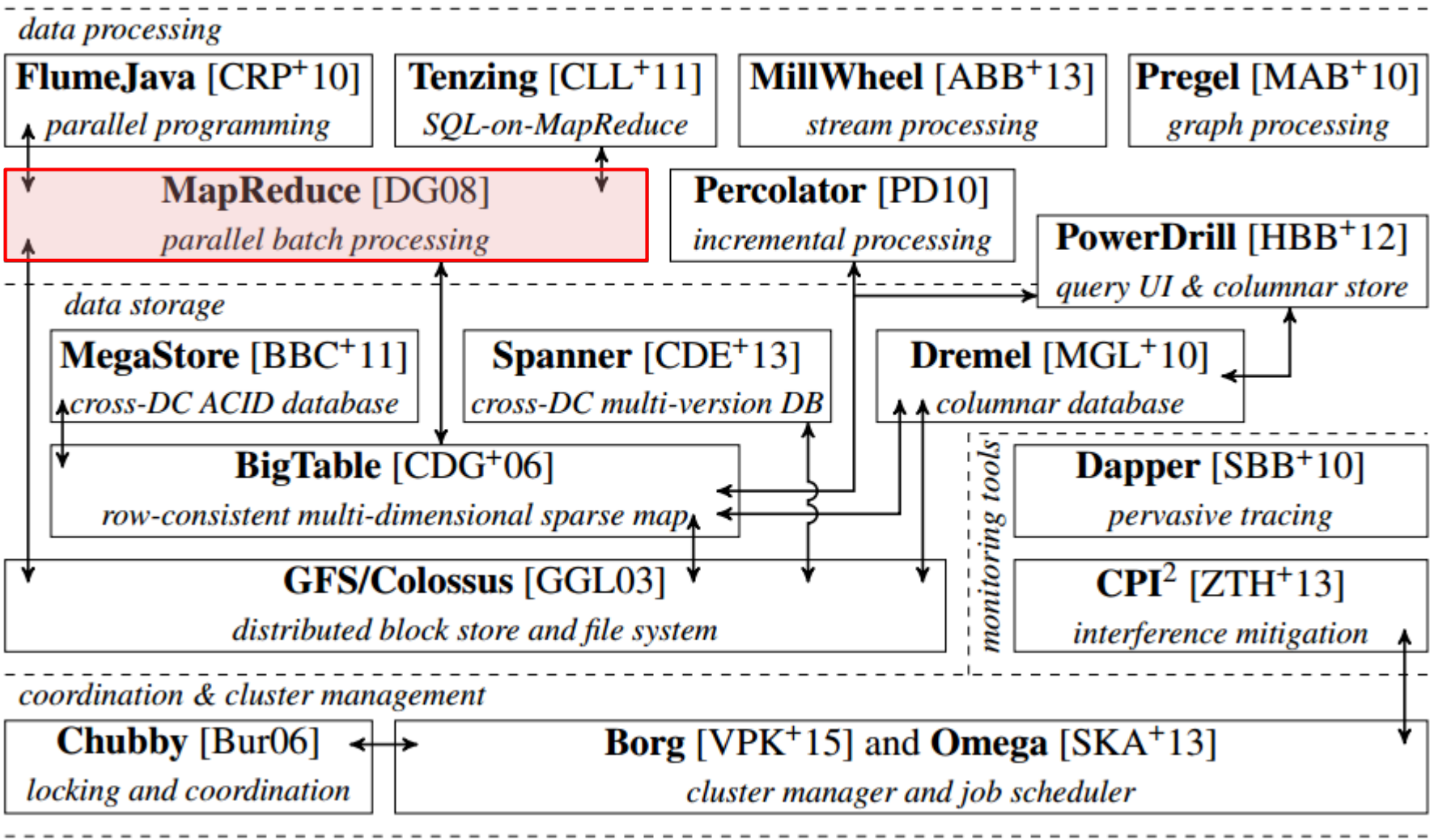
- Spanner offers **transactional** consistency: full RDBMS power, ACID properties, at global scale!

- Secret sauce: **hardware-assisted clock sync**
  - Using GPS and atomic clocks in data centres

- Use global timestamps and Paxos to reach consensus
  - Still have a period of uncertainty for write TX: **wait it out**!
  - Each timestamp is an **interval**:



The Google Stack



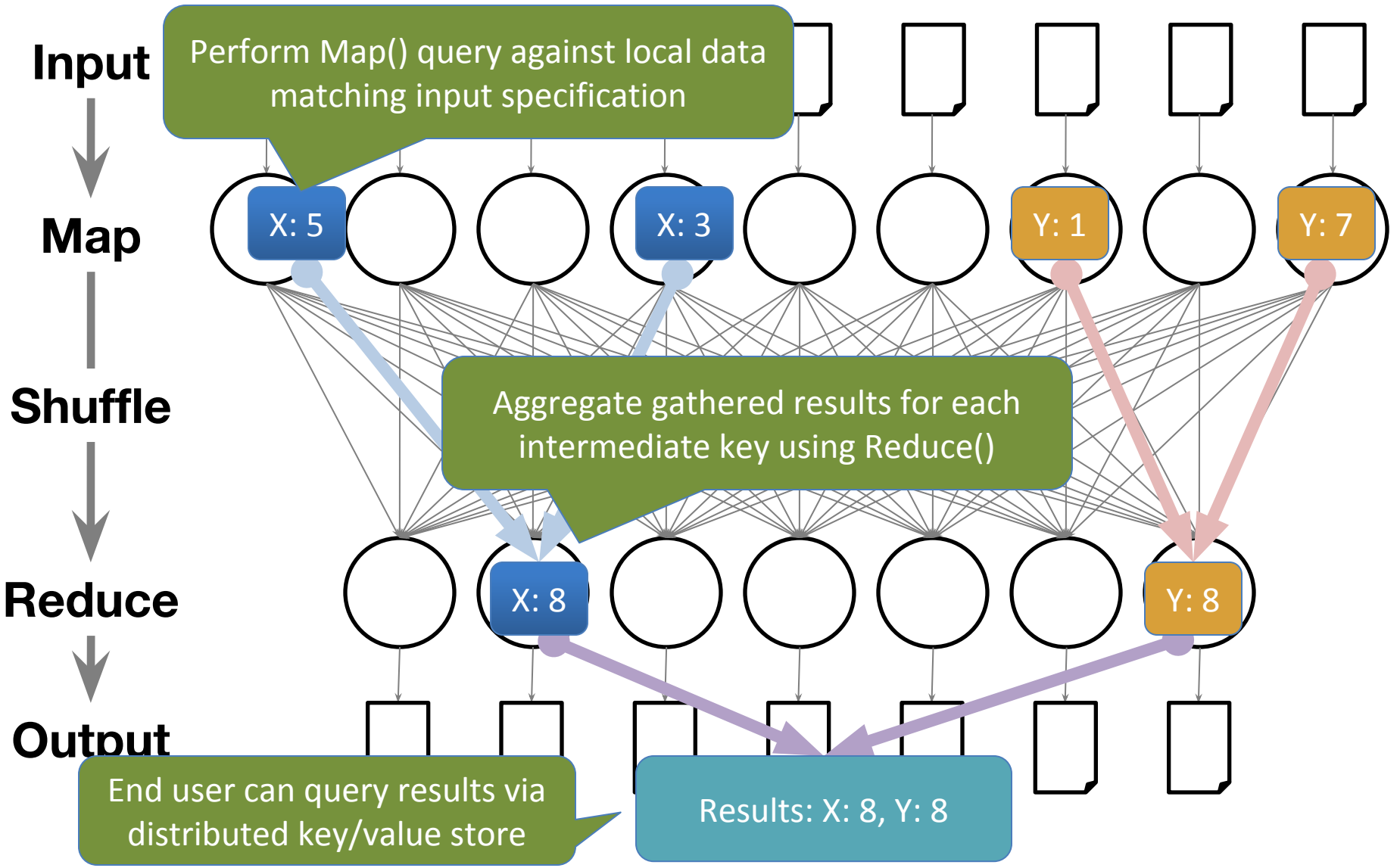
Details & Bibliography: http://malteschwarzkopf.de/research/assets/google-stack.pdf

Figure from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

MapReduce (2004)

- **Parallel programming framework** for scale
  - Run a program on 100's to 10,000's machines
- Framework takes care of:
  - Parallelization, distribution, load-balancing, scaling up (or down) & fault-tolerance
- **Accessible:** programmer provides two methods ;-)
  - map(key, value) → list of <key', value'> pairs
  - reduce(key', value') → result
  - Inspired by functional programming

MapReduce

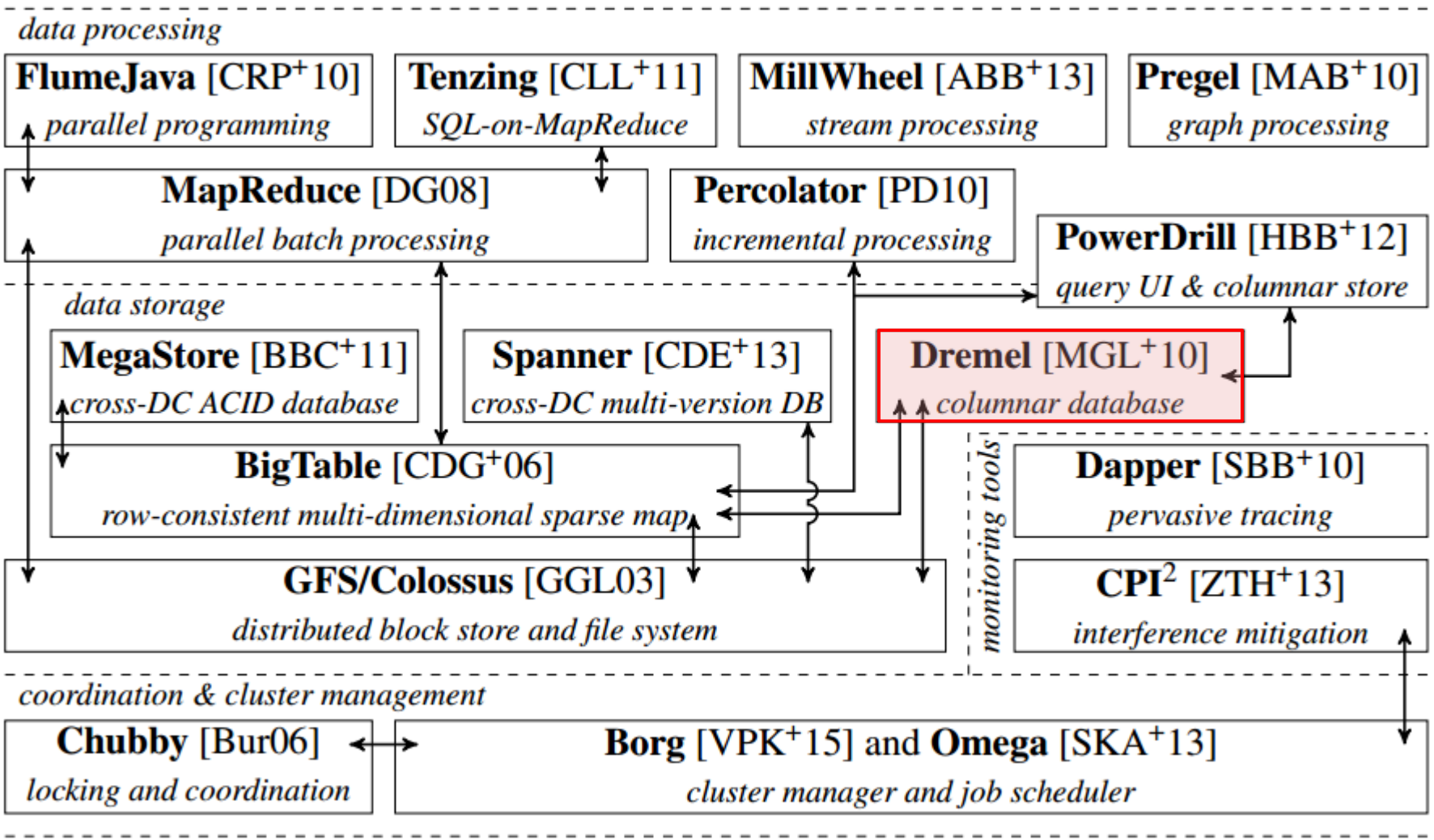


Slide originally due to S. Hand's distributed systems lecture course at Cambridge: http://www.cl.cam.ac.uk/teaching/1112/ConcDisSys/DistributedSystems-1B-H4.pdf

MapReduce: Pros & Cons

- **Extremely simple**, and:
  - Can **auto-parallelize** (since operations on every element in input are independent)
  - Can **auto-distribute** (since rely on underlying Colossus/BigTable distributed storage)
  - Gets **fault-tolerance** (since tasks are idempotent, i.e. can just re-execute if a machine crashes)
- Doesn't really use **any** sophisticated distributed systems algorithms (except storage replication)
- However, not a panacea:
  - Limited to batch jobs, and computations which are expressible as a `map()` followed by a `reduce()`

The Google Stack

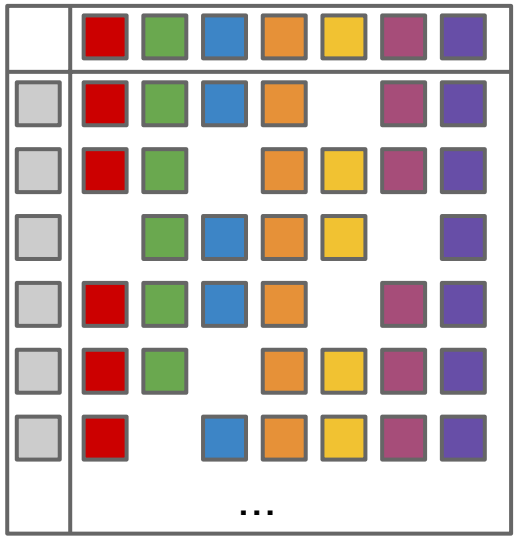


Details & Bibliography: http://malteschwarzkopf.de/research/assets/google-stack.pdf

Figure from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

Dremel (2010)

- **Column-oriented** store
  - For quick, interactive queries



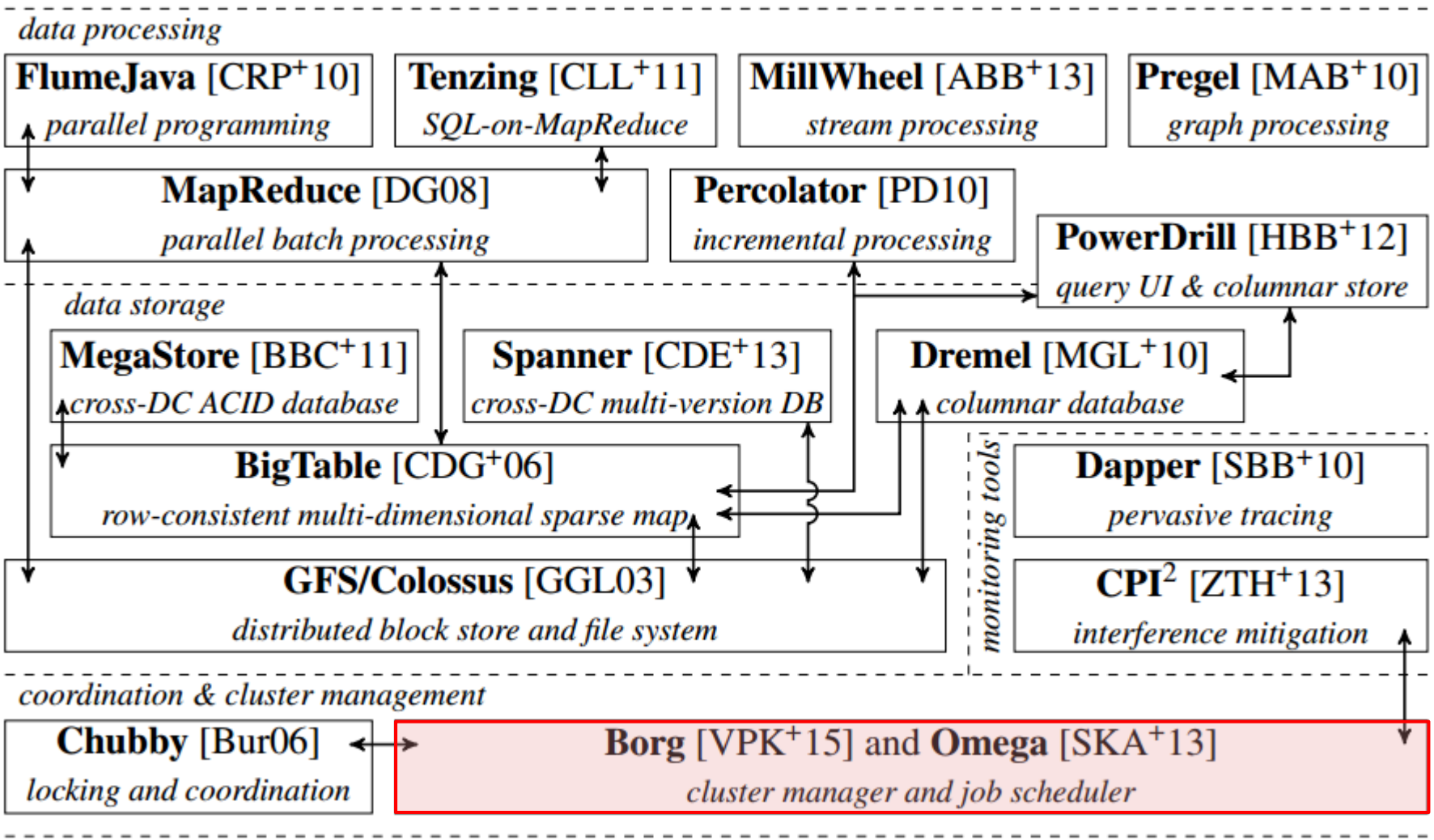
Row-oriented storage

Column-oriented storage

Dremel (2010)

- Stores **protocol buffers**
  - Google's universal serialization format
  - Nested messages → nested columns
  - Repeated fields → repeated records
- Efficient encoding
  - Many **sparse records**: don't store NULL fields
- Record re-assembly
  - Need to put results back together into records
  - Use a **Finite State Machine** (FSM) defined by protocol buffer structure

The Google Stack



Details & Bibliography: http://malteschwarzkopf.de/research/assets/google-stack.pdf

Figure from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

Borg/Omega

- **Cluster manager** and scheduler
  - Tracks machine and task liveness
  - Decides where to run what
- Consolidates workloads onto machines
  - Efficiency gain, cost savings
  - Need fewer clusters

Borg/Omega

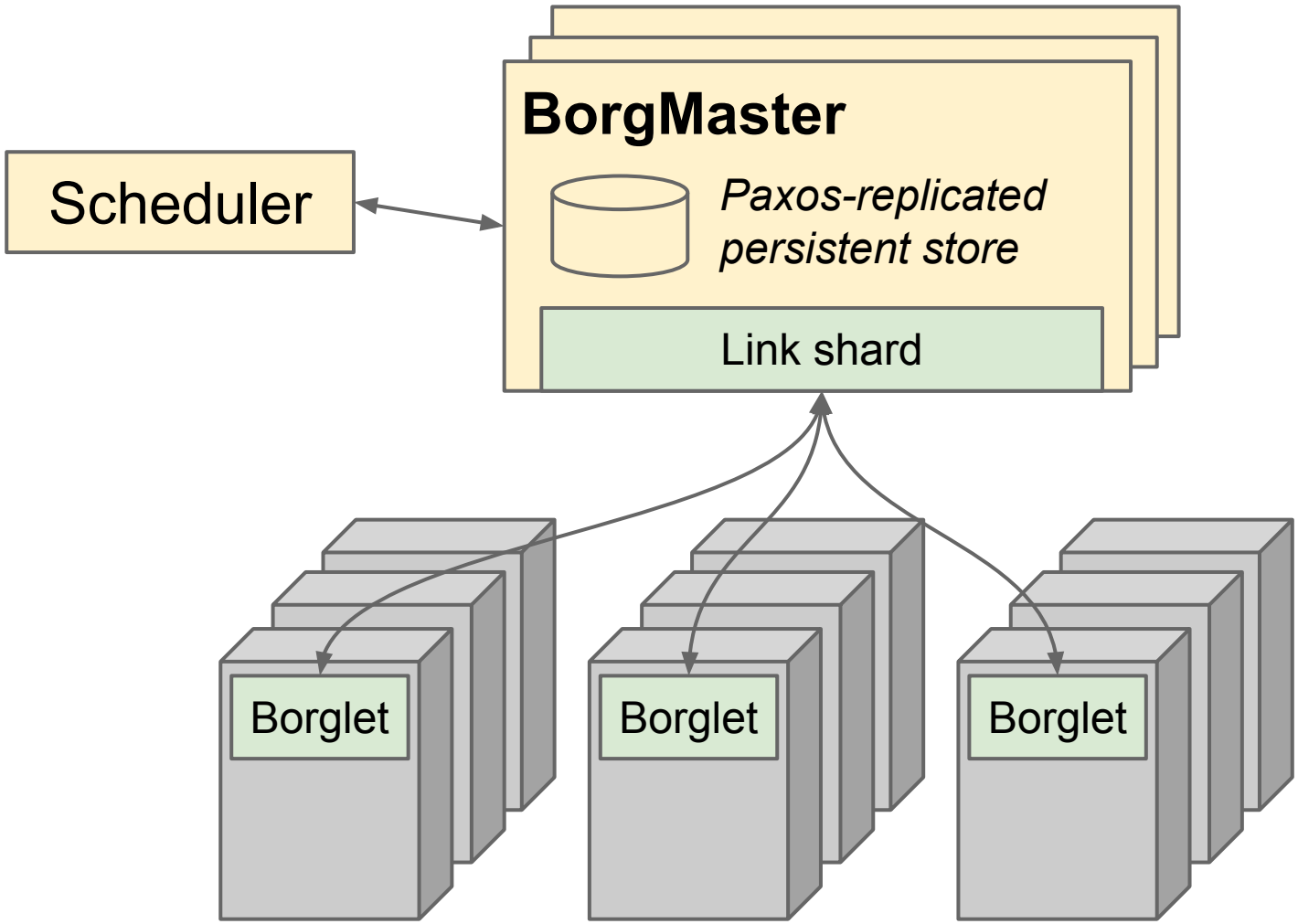


Figure reproduced after A. Verma et al., "Large-scale cluster management at Google with Borg", Proceedings of EuroSys 2015.

Borg/Omega: workloads

Cluster A

Medium size

Medium utilization

Cluster B

Large size

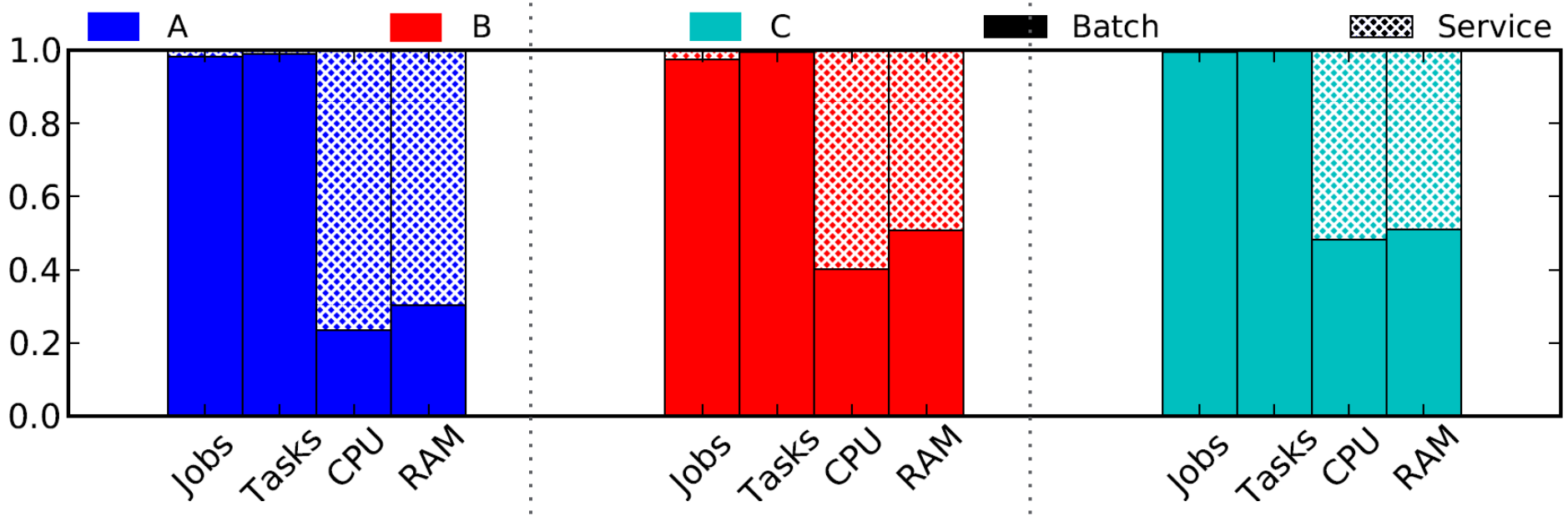
Medium utilization

Cluster C

Medium (12k mach.)

High utilization

Public trace



Jobs/tasks: counts

CPU/RAM: resource seconds [i.e. resource * job runtime in sec.]

Figure from M. Schwarzkopf et al., "Omega: flexible, scalable schedulers for large compute clusters", Proceedings of EuroSys 2013.

Borg/Omega: workloads

Service jobs run for much longer than batch jobs:
long-term user-facing services vs. one-off analytics.

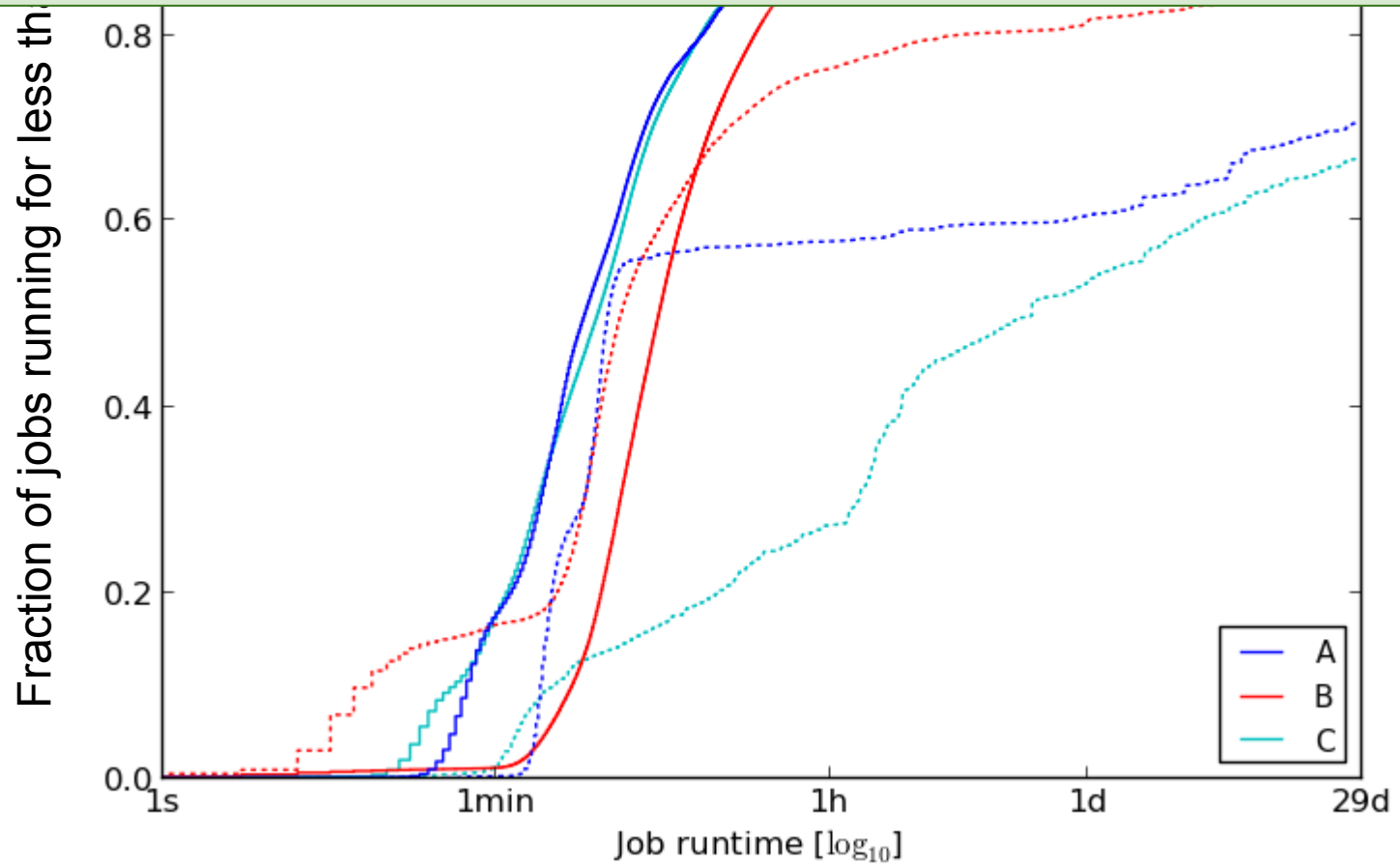


Figure from M. Schwarzkopf et al., "Omega: flexible, scalable schedulers for large compute clusters", Proceedings of EuroSys 2013.

Borg/Omega: workloads

Batch jobs arrive more frequently than service jobs:
more numerous, shorter duration, fail more.

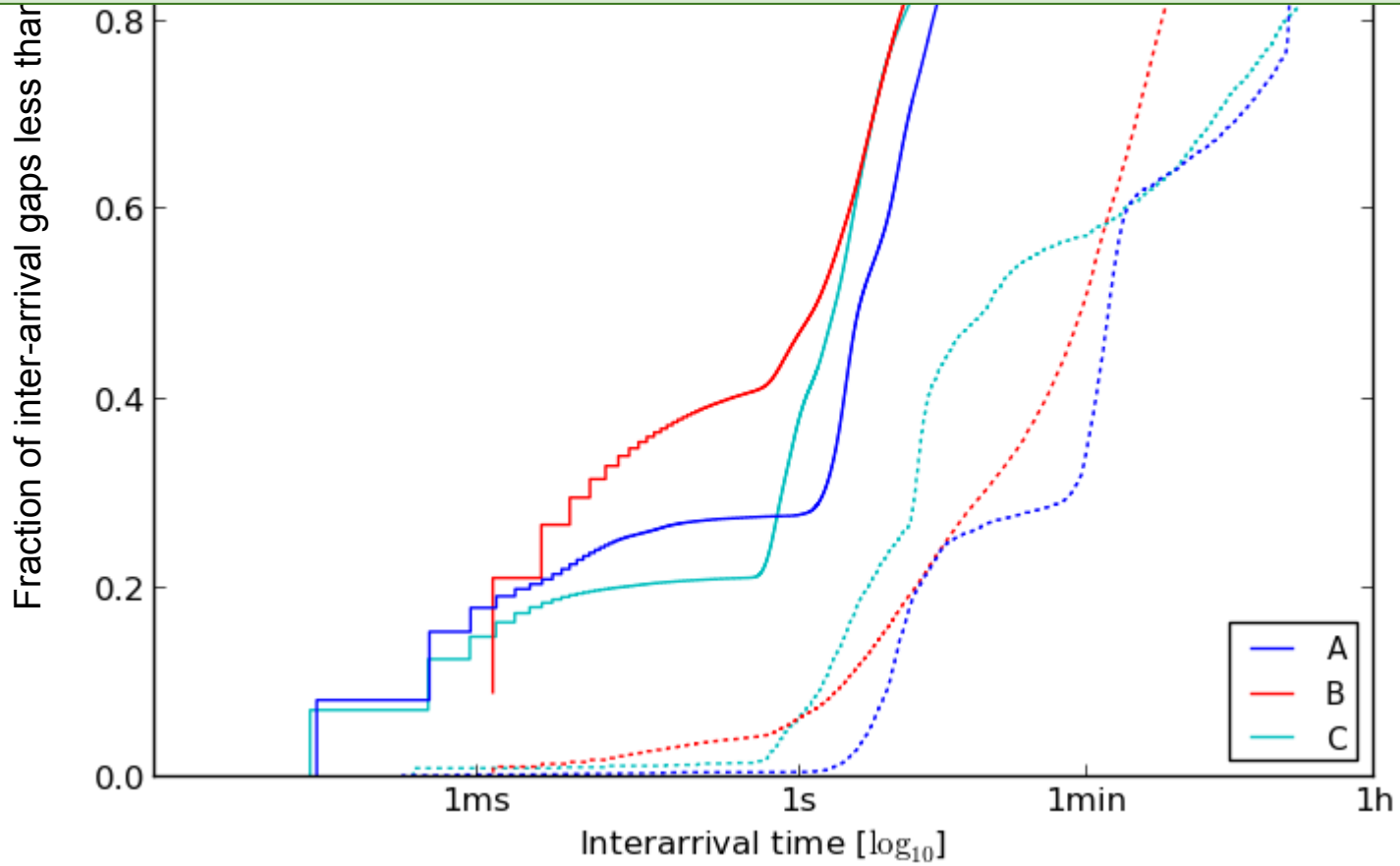
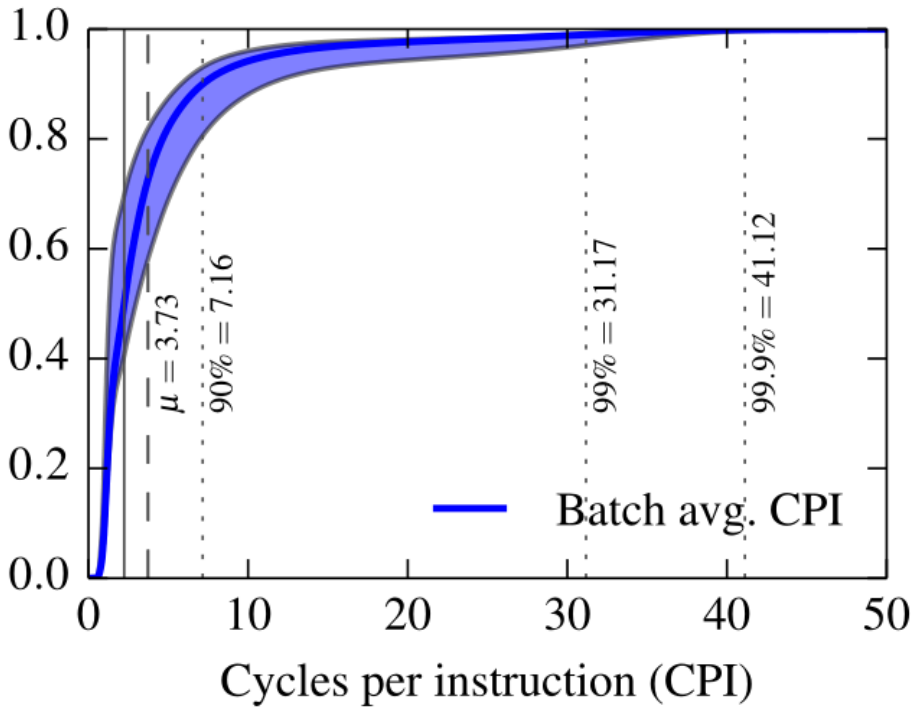
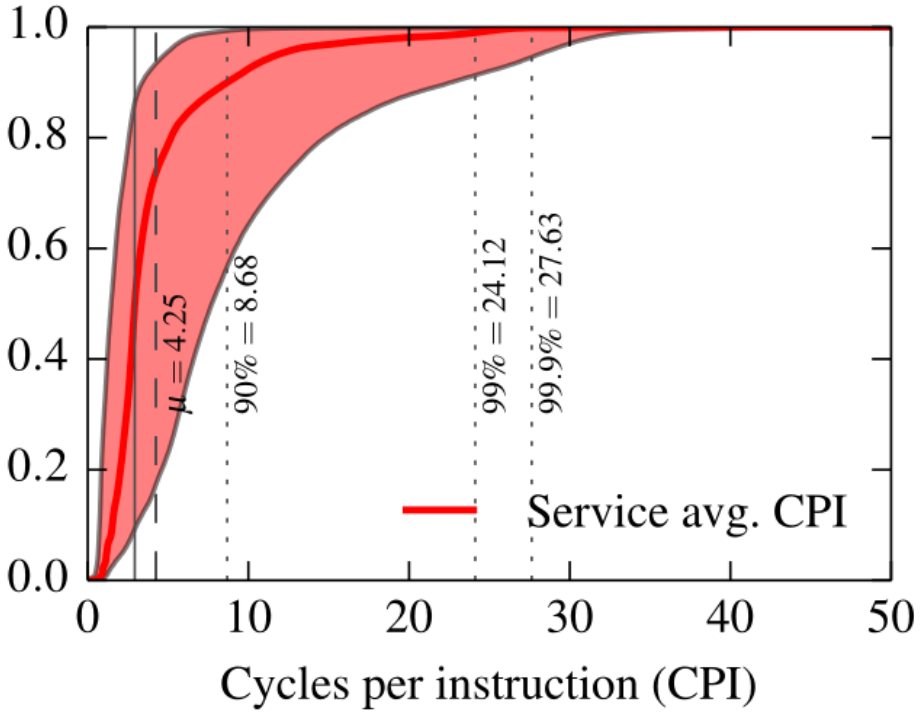


Figure from M. Schwarzkopf et al., "Omega: flexible, scalable schedulers for large compute clusters", Proceedings of EuroSys 2013.

Borg/Omega: workloads

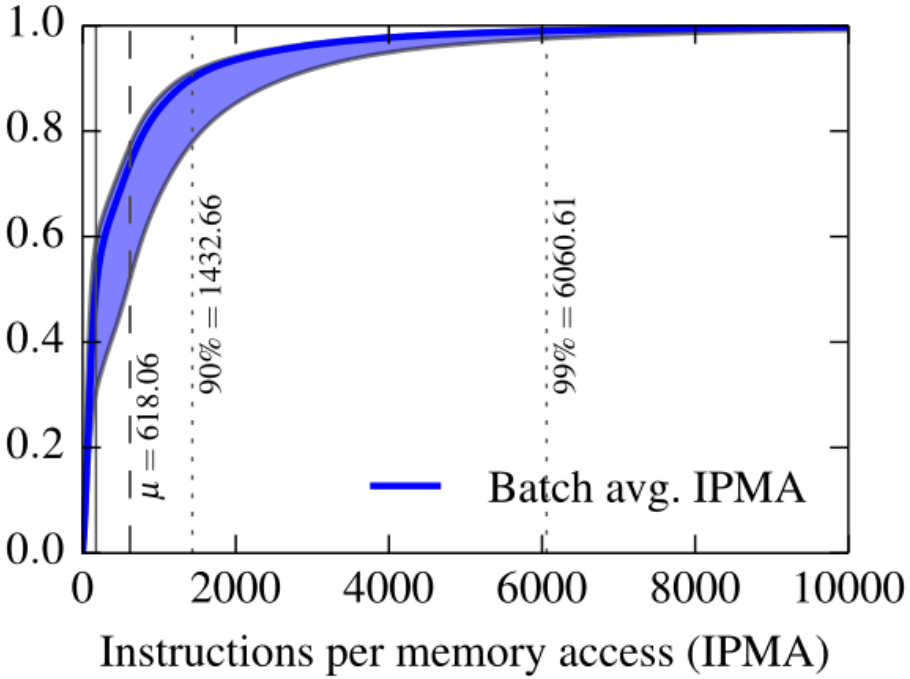


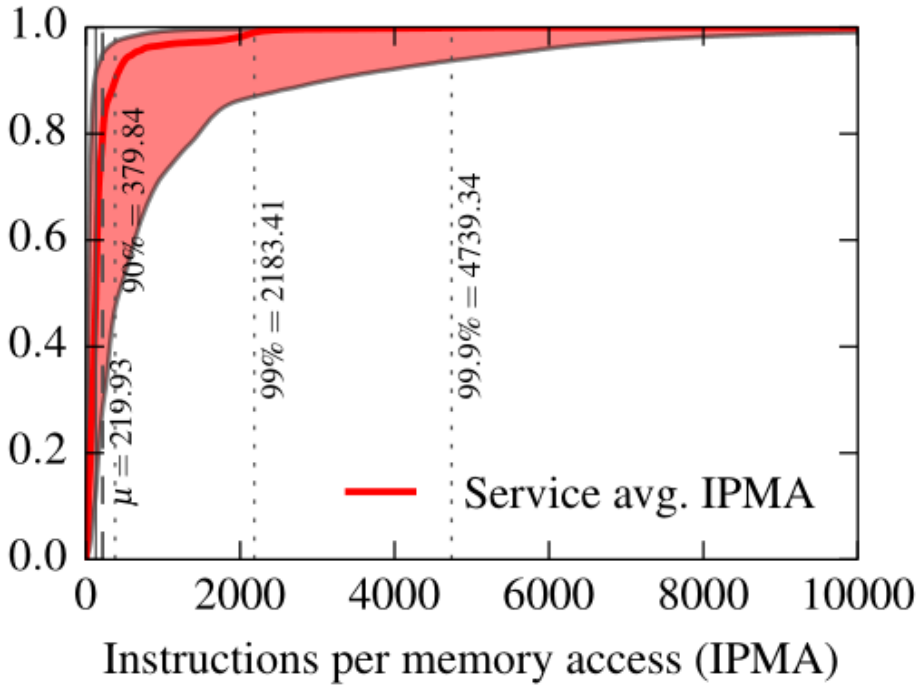


Batch jobs have a longer-tailed CPI distribution:
lower scheduling priority in kernel scheduler.

Figures from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

Borg/Omega: workloads





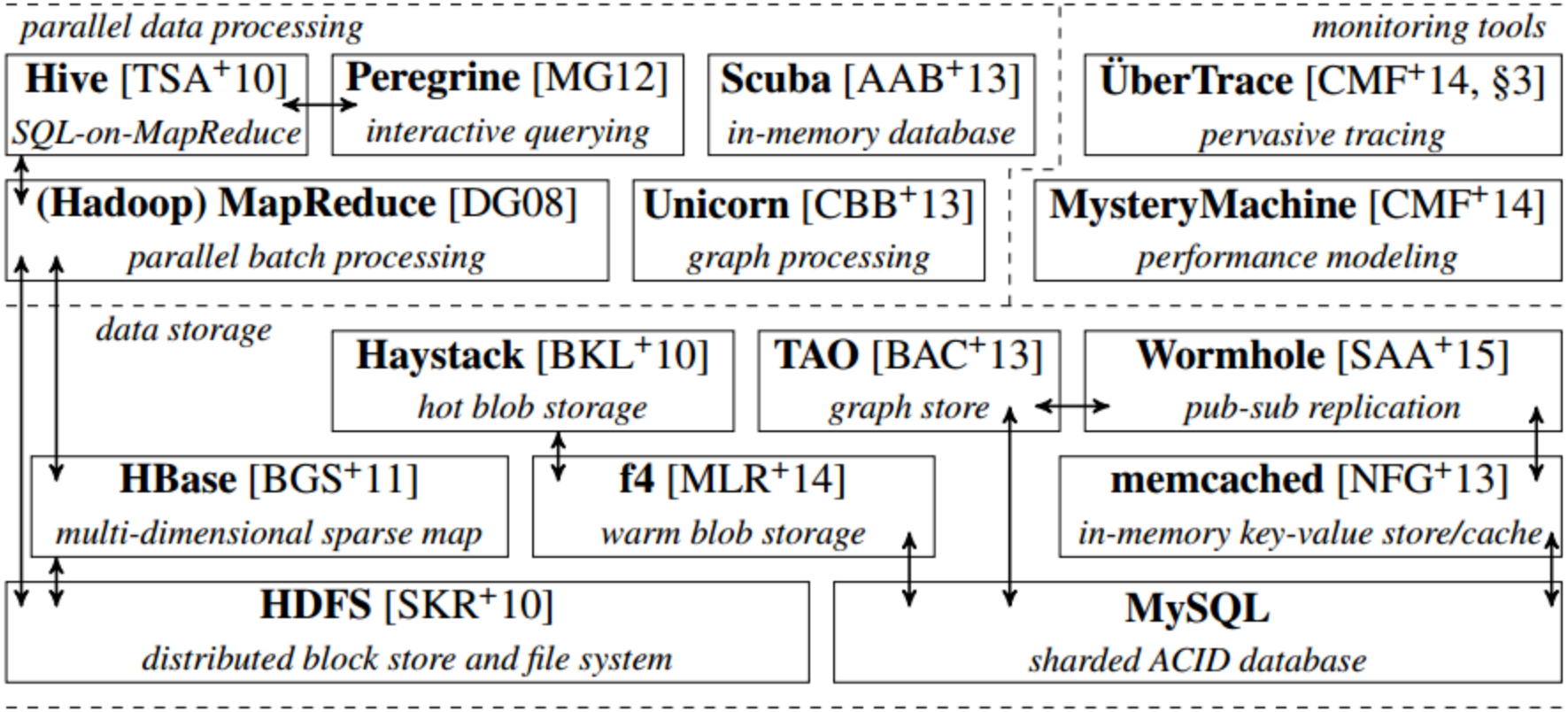
Service workloads access memory more frequently:
larger working sets, less I/O.

Figures from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

How's this different to HPC?

- Generally **avoid tight synchronization**
  - No barrier-syncs!
  - Asynchronous replication, eventual consistency
- Much more **data-intensive**
  - Lower compute-to-data ratio
- Aggressive **resource sharing** for utilization

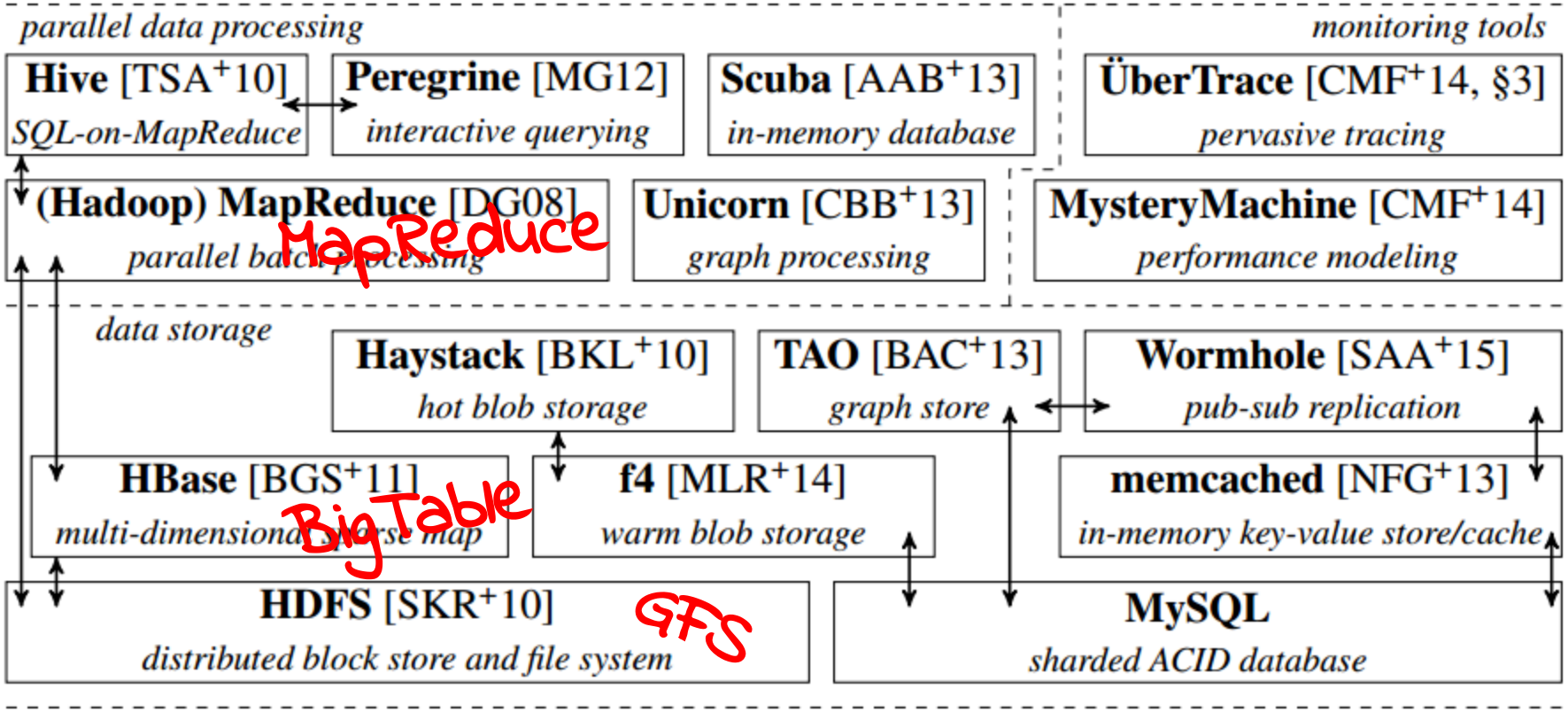
The facebook Stack



Details & Bibliography: http://malteschwarzkopf.de/research/assets/facebook-stack.pdf

Figure from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

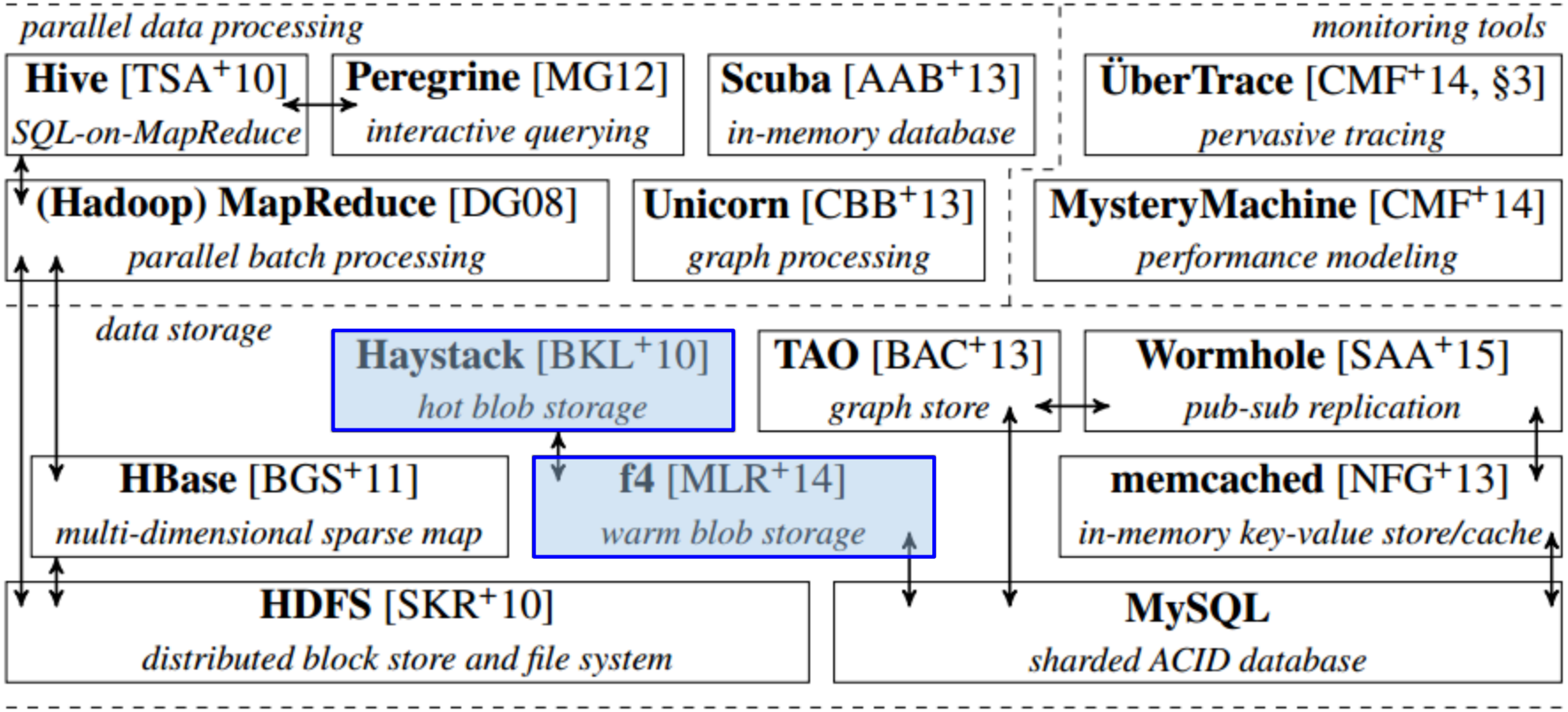
The facebook Stack



Details & Bibliography: http://malteschwarzkopf.de/research/assets/facebook-stack.pdf

Figure from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

The facebook Stack



Details & Bibliography: http://malteschwarzkopf.de/research/assets/facebook-stack.pdf

Figure from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

Haystack & f4

- **Blob stores**, hold photos, videos
  - **not:** status updates, messages, like counts
- Items have a level of **hotness**
  - How many users are currently accessing this?
  - Baseline “cold” storage: MySQL
- Want to **cache close to users**
  - Reduces network traffic
  - Reduces latency
  - But cache capacity is limited!
  - Replicate for performance, not resilience

What about other companies' stacks?

How about other companies?

- Very similar stacks.
  - Microsoft, Yahoo, Twitter all similar in principle.
- Typical set-up:
  - Front-end serving systems and fast back-ends.
  - Batch data processing systems.
  - Multi-tier structured/unstructured storage hierarchy.
  - Coordination system and cluster scheduler.
- Minor differences owed to business focus
  - e.g., Amazon focused on inventory/shopping cart.

Open source software

Lots of open reimplementations!

- MapReduce → Hadoop, Spark, Metis
- GFS → HDFS
- BigTable → HBase, Cassandra
- Borg/Omega → Mesos, Firmament

But also some releases from companies…

- Presto (Facebook)
- Kubernetes (Google)

Current research

- Many hot topics!
  - This distributed infrastructure is still new.
- Make many-core machines go further
  - Increase utilization, schedule better.
  - Deal with co-location interference.
- Rapid insights on huge datasets
  - Fast, incremental stream processing, e.g. Naiad.
  - Approximate analytics, e.g. BlinkDB.
- Distributed algorithms
  - New consensus systems, e.g. RAFT.

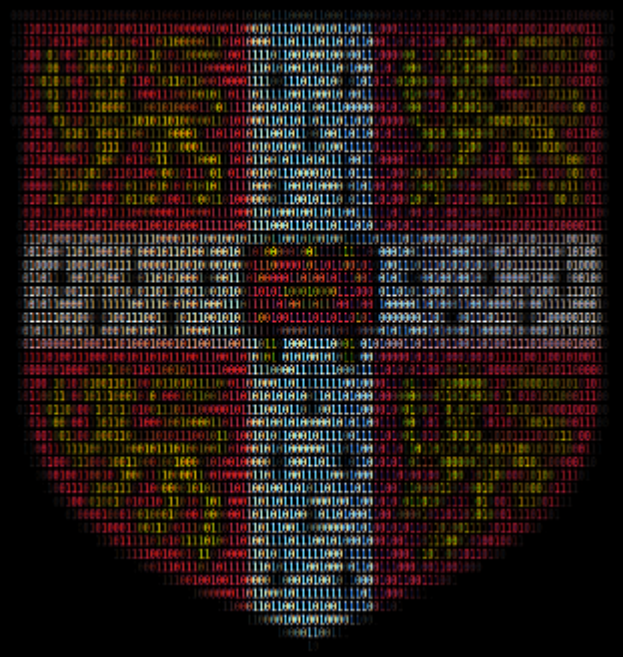
Conclusions

1. Running at huge (10k+ machines) scale requires different software stacks.

2. Pretty interesting systems
   a. Do read the papers!
   b. (Partial) Bibliography:

      http://malteschwarzkopf.de/research/assets/google-stack.pdf
      http://malteschwarzkopf.de/research/assets/facebook-stack.pdf

3. Lots of activity in this area!
   a. Open source software
   b. Research initiatives



Cambridge Systems at Scale

@CamSysAtScale

camsas.org