Scalability in Semantic Data Management

Philippe Cudré-Mauroux
[eXascale Infolab]
U. of Fribourg—Switzerland

April 23, 2012 // Dagstuhl Seminar on Semantic Data Management

Forewarning

- Scalability
  - Multifaceted, extremely challenging issue
    - ongoing efforts for much simpler data models than SW models
  - *Critical* for large-scale problems
    - LOD / Enterprise data

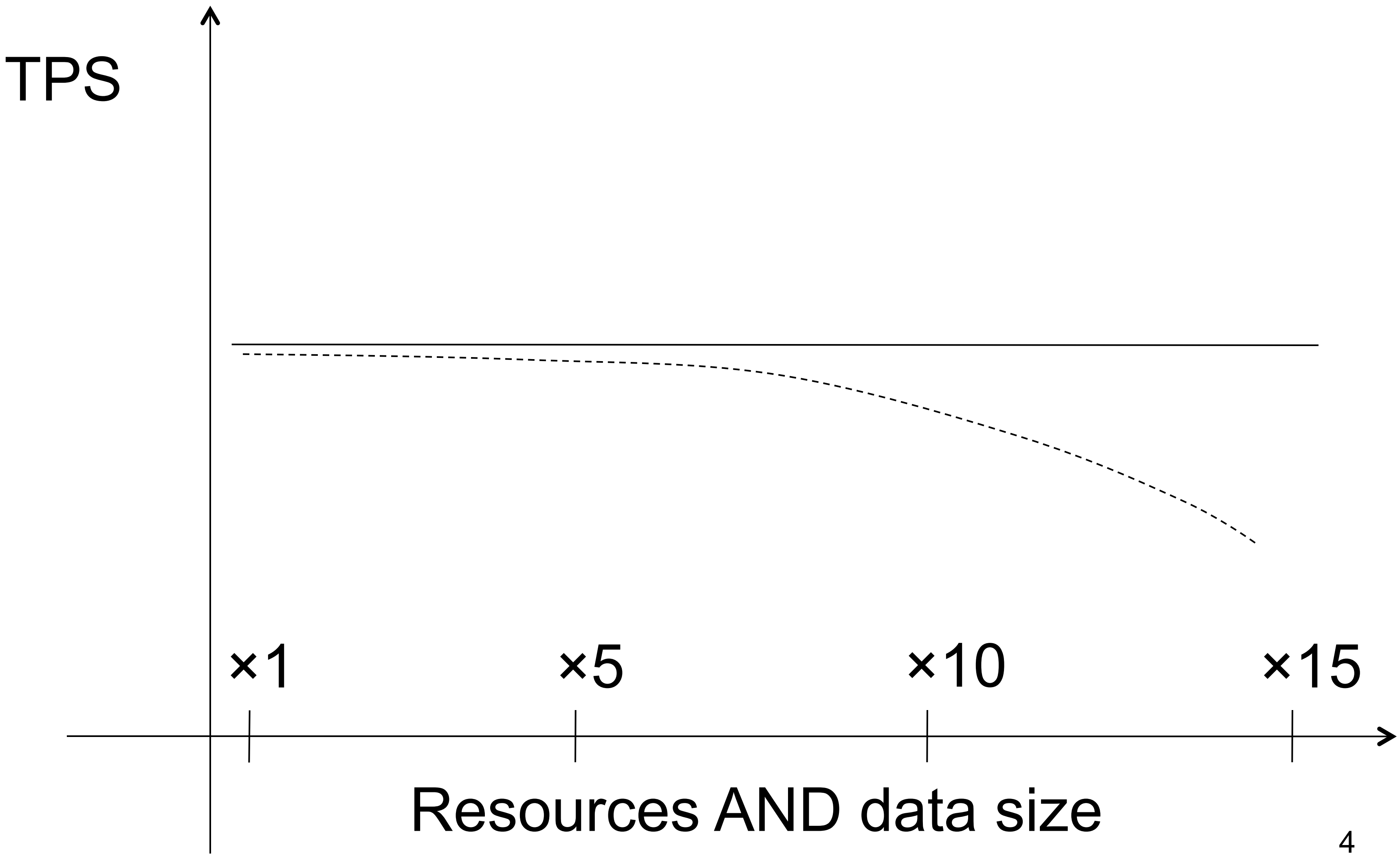
- Tutorial?
  - One (simplified) story

- Bias ahead towards
  - My background (Decentralized P2P & OSDNFA DB *systems*)
  - The metrics and problems I am interested in (e.g., TPS for complex queries on massive graphs)

Scalability

- [Wikipedia] “scalability is the ability of a system, network, or process, to handle growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth.”

Linear v.s. Non-linear Scaleup



Scale up VS scale out

- Scale up
  - Vertical scaling
  - Add resources to a single node or system

- Scale out
  - Horizontal scaling
  - Add more nodes to the system

Scaling Up SW Systems

- Throwing resources at a problem is only effective when addressing the bottleneck of the system
  - CPU-bound contexts
  - IO-bound contexts (disk / network-bound)

  $\Rightarrow$Better CPU $\neq$ faster data processing

- Bottleneck depends on
  - System
  - Hardware
  - Workload

Shifting Bottlenecks

- Bottlenecks are thus not always application-dependent
  - Advances, CPU/disk tradeoffs
  - Ex.: complex analytics used to be disk-bound on legacy RDMBSs, but Vertica is often CPU-bound

⇒Importance of (good) benchmarking

⇒Various benchmarks

⇒Benchmarking platforms

Scaling-up Disk-bound SW Systems

- Disk is often the bottleneck because of
  - Slow buffered reads (e.g., 50MB/s)
  - *Slower than slow* random reads (seek = 10ms)

- (At least) four ways of scaling the issue
  - Read less (e.g., compress, index, cache)
  - Jump less (e.g., co-locate)
  - Buy new disks
  - Climb up the memory hierarchy
    - Study Flash / PCM / Main-memory characteristics

Recycling Old Technology

- Typically yields suboptimal results (unless the context has not changed)
  - N-ary storage: very verbose
  - B-trees: often require many seeks
  - Relational model: implies many self-joins

A few SW examples

- Native indexing: Hexastore [Weiss08]
- Native indexing + compression and a bunch of neat tricks: RDF-3x [Neumann08]
- Co-location + compression: Vertical SW-Store [Abadi07]
- Co-location, co-location, co-location: dipLODocus[RDF] [Wylot11]
- Further examples?

What about Main-Memory Systems?

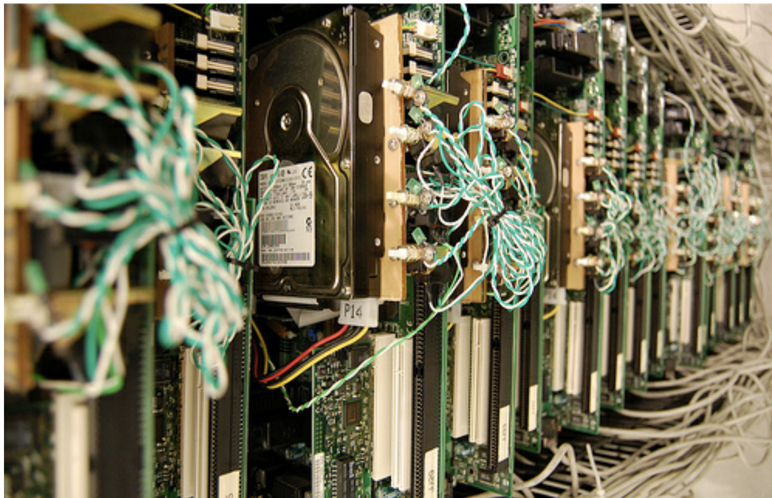
- (Hyrise system [Grund10]) Distinction between
  - Layout-independent operations (e.g., mathematical operations)
  - Layout-dependent operations (e.g., reads, writes…)

- Layout-independent operations
  - New physical operators, algorithms
  - Multi-core optimizations

- Layout-dependent operations
  - Cache locality!
  - Many of the techniques from the previous slide can apply

Scaling out SW Systems

- Historically, three architectures of parallel DBMSs
  - Shared Memory
  - Shared Disk
  - Shared Nothing
- Expensive to scale shared memory / disks systems
- Shared nothing is prevalent today
  - Clusters of commodity machines (Google, Hadoop, AWS, SciDB...)
    - Simply add boxes / racks
    - Failure is the norm
  - Also the most complex to design…

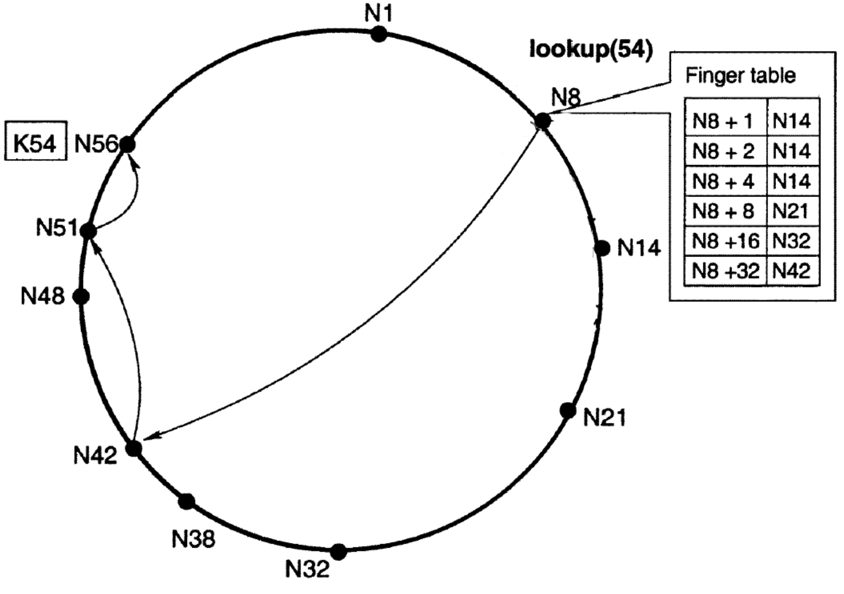


Contrasted Scenarii

- Typical high-throughput, read-write OLTP workloads
  - Load-balancing, inter-query parallelism
  - Functional redundancy
  - Logging, consistency (CAP)
- Typical complex analytic workloads
  - Intra-operator parallelism
  - Amdahl's law
  - Data-to-node affinity
- Current and future workloads for SW systems?
  - From read-mostly to read-write Web?

Typical SW Scale-out

- GridVine (hash-partitioning) [Aberer04]
  - Index subject, predicate and object in a distributed identifier space
  - Queries are resolved using iterative, distributed look-ups + joins
  - GridVine based on a DHT; now alternate (better?) substrates
    - Key-value stores
    - Hadoop
    - Others?



Example of Query Resolution

– Select ?s Where {
  ?s is_a Student
  ?s lives_in Seattle
  ?s takes ?c
  ?c is_a GraduateCourse }

– $\pi_s\ \sigma_{p=\text{"is\_a"}\ \wedge\ o=\text{"Student"}}$
$\bowtie\ \pi_s\ \sigma_{p=\text{"lives\_in"}\ \wedge\ o=\text{"Seattle"}}$
$\bowtie\ \pi_s\ (\sigma_{p=\text{"takes"}}\ {}_{o}{\bowtie}_{s}\ \ \sigma_{p=\text{"is\_a"}\ \wedge\ o=\text{"GraduateCourse"}})$

Better Partitionings?

- Triple-table indexing requires one distributed look-up per triple pattern + join
- Often leads to poor intra-query parallelism

- Better (graph) partitionings?
  - Range partitioning?
  - K-Means? [Huang11]
  - Full graph decomposition? [Microsoft Horton]
  - RDF molecules? [dipLODocus]
  - Other examples?

Yet Another Problem worth Tackling

- Federated (LoD) queries
  - Collaborative queries involving independent nodes
  - Fairly new actually
    - Not your typical wrapper-mediator system
- Several interesting research directions for scalability
  - FedX [Schwarte2011]
    - Sesame + efficient distributed processing
  - Graph optimizations [Wang2011]
  - Collaborative *memcached*?
  - Further directions?

Other Scalability Aspects I Omitted

- SW Scalability w.r.t.
  - Reasoning
  - Search & ranking
  - Entity resolution
  - Data ingestion
  - Data cleaning
  - Data linking
  - Data provenance
  - Etc.

- Looking forward to discussing some of these this week

(Some) Conclusions

- SW Scalability is a multifaceted and challenging problem
  - *Essential* for many SW deployments
- Recycling old techniques often lead to sub-optimal solutions
  - Unless the context is similar
- Building dedicated platforms is generally-speaking a good idea
  - Requires a good understanding of hardware platform and workload
  - Benchmark and benchmarking infrastructures, e.g. BowlognaBench, http://oltpbenchmark.com

References (1)

- [Weiss08] Cathrin Weiss, Panagiotis Karras, and Abraham Bernstein. 2008. Hexastore: sextuple indexing for semantic web data management. Proc. VLDB Endow. 1, 1.
- [Neumann08] Thomas Neumann and Gerhard Weikum. 2008. RDF-3X: a RISC-style engine for RDF. Proc. VLDB Endow. 1, 1.
- [Abadi07] Daniel J. Abadi, Adam Marcus, Samuel R. Madden, and Kate Hollenbach. 2007. Scalable semantic web data management using vertical partitioning. In Proceedings of the 33rd international conference on Very large data bases (VLDB '07)
- [Wylot11] Marcin Wylot, Jigé Pont, Mariusz Wisniewski, and Philippe Cudré-Mauroux. 2011. dipLODocus[RDF]: short and long-tail RDF analytics for massive webs of data. In Proceedings of the 10th international conference on The semantic web (ISWC'11).
- [Grund10] Martin Grund, Jens Krüger, Hasso Plattner, Alexander Zeier, Philippe Cudré-Mauroux, Samuel Madden: HYRISE - A Main Memory Hybrid Storage Engine. PVLDB 4(2): 105-116 (2010).

References (2)

- [Aberer04] Karl Aberer, Philippe Cudré-Mauroux, Manfred Hauswirth, Tim Van Pelt: GridVine: Building Internet-Scale Semantic Overlay Networks. International Semantic Web Conference (ISWC 2004)

- [Huang2011] Jiewen Huang, Daniel J. Abadi, Kun Ren: Scalable SPARQL Querying of Large RDF Graphs. PVLDB 4(11): 1123-1134 (2011)

- [Schwarte2011] Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, Michael Schmidt: FedX: A Federation Layer for Distributed Query Processing on Linked Open Data. ESWC (2) 2011: 481-486

- [Wang2011] X. Wang, T. Tiropanis, and H. Davis, “Evaluating graph traversal algo-rithms for distributed sparql query optimization,” in Joint International Semantic Technology Conference (JIST2011), 2011.