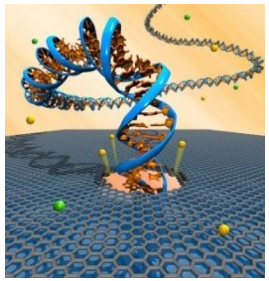
Elastic NoSQL databases over the Cloud

I. Konstantinou, E. Angelou, C. Boumpouka,
D. Tsoumakos, N. Koziris

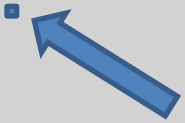
Computing Systems Laboratory
School of Electrical and Computer Engineering
National Technical University of Athens

Big Data

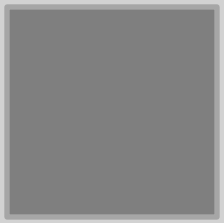
- '*Moore's*' Law: Data doubles every 18 months
- 90% of today's data was created in the last 2 years
  - Facebook: 20TB/day compressed
  - CERN/LHC: 40TB/day (15PB/year)
  - NYSE: 1TB/day
- Many more
  - Web logs, financial transactions, medical records, etc



Data Growth



1 EB (Exabyte-10^{18}) = 1000 PB (Petabyte-10^{15})
Last year (2010) US mobile data traffic



0.8 ZB (Zettabyte) = 800 EB
Entire global mass of digital data in 2009
according to IDC

35 ZB (Zettabyte-10^{21})
IDC's forecast for all digital data in 2020

Cloud Computing

- Resource provisioning “as a service”
  - CPUs, Disks, networks, developing platforms, applications, etc
  - Virtualized resources from distant data centers
- Charging model
  - “Pay as you go” model
  - OPEX instead of CAPEX
- Management model
  - Elasticity
  - Easy resource manipulation according to application needs
- Enterprise driven
  - Amazon, Google, IBM, Microsoft, etc

Motivation – the story(1)

- ‘Big-data’ processing era
  - (Web) analytics, science, social, business
  - Store + analyze everything
- Distributed, high-performance processing
  - From P2P to Grid computing
  - And now to the clouds…

- Traditional databases not up to the task
  - NoSQL databases

Motivation – the story (2)

- NoSQL
  - Non-relational
  - Horizontal scalable
  - Distributed
  - Open source
  - And often:
    - schema-free, easily replicated, simple API, eventually consistent /(not ACID), big-data-friendly, etc
  - Many, many, implementations…
    - Google's BigTable, Facebook's Cassandra, LinkedIn's Voldermort, MongoDB,
  - 120+ implementations, http://nosql-database.org/

NoSQLs + elasticity

- Column family
  - **Hbase, Cassandra**, …
- Document store
  - CouchDB, mongoDB, …
- Key-Value store
  - **Riak**, Dynamo, Voldemort, …
- Many offer **elasticity+sharding:**
  - Expand/contract resources according to demand
  - Pay-as-you-go, robustness, performance
  - Shared-nothing architecture allows that
  - Important! See recent foursquare and netflix outage
- Isn't that what PaaS offers?

thus…(end of the story)

- PaaS and NoSQLs are (or should be) inherently elastic
- How efficiently do they implement elasticity?
  - NoSQLs over an IaaS platform
    - (EC2, **Eucalyptus**, OpenStack)
  - No study that registers qualitative + quantitative results
- Related
  - Report NoSQL performance (not elasticity)
  - Cloud platform elasticity (no NoSQL)
  - Domain-specific

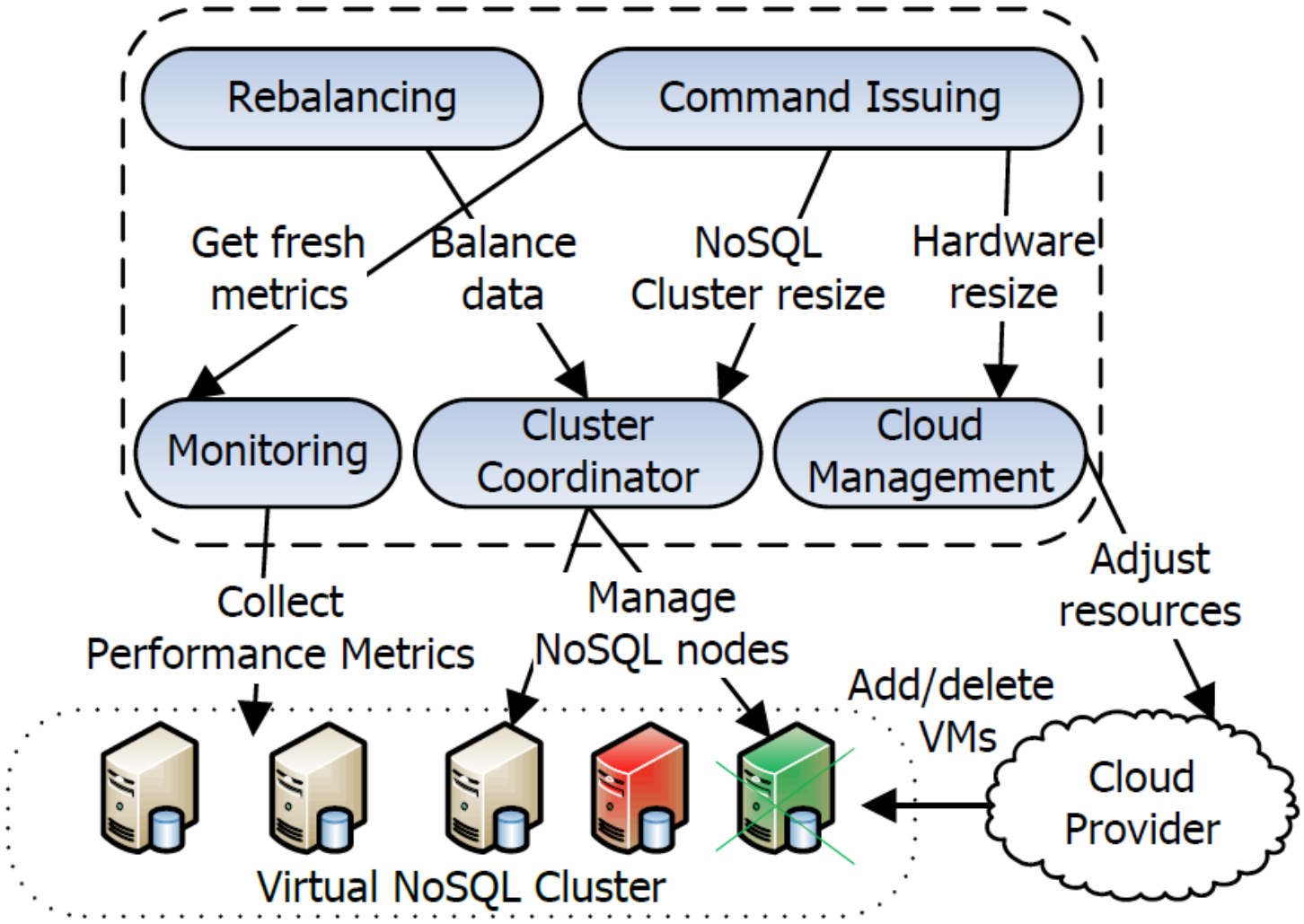
Contribution (1)

- VM-based framework for NoSQL cluster monitoring
- For a cluster resize, identify and measure
  - Cost, gains
  - In terms of:
    - Time, effort, increase in throughput, latency, …?
- Ultimate goal: Provide a generic platform
  - **any** NoSQL engine
  - User-defined policies
  - Automatic resource provisioning
- Example towards this goal
  - Tiramola

Contribution (2)

- Coding + infrastructure
  - 2K lines open source python code (GFOSS + google code)
  - **http://tiramola.googlecode.com**
- Using cloud-based client tools, platform-agnostic
  - EucaTools guarantee execution in numerous cloud platforms
- Cassandra, Hbase and Riak implementation
  - almost Voldemort
- How-to, best practices, glitches, erroneous assumptions, …

Framework architecture



National Technical University of Athens CSLab

Architectural considerations

- Robustness
  - Daemon process that checkpoints and can be restarted
  - State is provided from the IaaS Cloud and the Monitoring module.
  - Applicable timeouts (not realtime systems!)
- Modularity
  - Different interchangeable components
  - APIs that utilize primitives (NoSQL and Policies)
- Expandability
- Speed
- Written in Python

Platform Setup

- 16 physical nodes
  - 2xQuadCore E5520 Intel Xeon® Hyperthreading (@2.27Ghz)
  - 48GB RAM, 2 SAS – RAID 1
- Virtual Machines
  - Similar to an Amazon EC2 large instance
  - 4-core processor, 8GB RAM, 50GB disk space
  - QCOW image: 1.6GB compressed, 4.3GB uncompressed
    - Available for download from googlecode
  - VM root fs instead of EBS (Netflix outage)
- Cluster
  - Eucalyptus 2.0.0 with dedicated Cloud/cluster controller

Experiments overview

- Indentify which DB metrics are affected under various loads
  - Consider both server-side and client-side metrics
- Identify costs + gain for a cluster resize
  - Cost of adding/removing nodes
  - Gains of increasing cluster size (how many nodes?)
- Check automated cluster resize
  - Using Hbase

Cluster Resize Time considerations

- VM initialization
  - 3min for addition, negligible for removal (few secs)
- Node configuration
  - Config files and propagation (at most 30 sec cycle)
- Region rebalance
  - Actively participate in the NoSQL cluster
  - Cassandra more efficient, Hbase depends on data, #nodes,…
- Data rebalance
  - Optional
  - Hbase: data / cluster-size dependent (+2h)
  - Cassandra: individual `loadbalance` signals

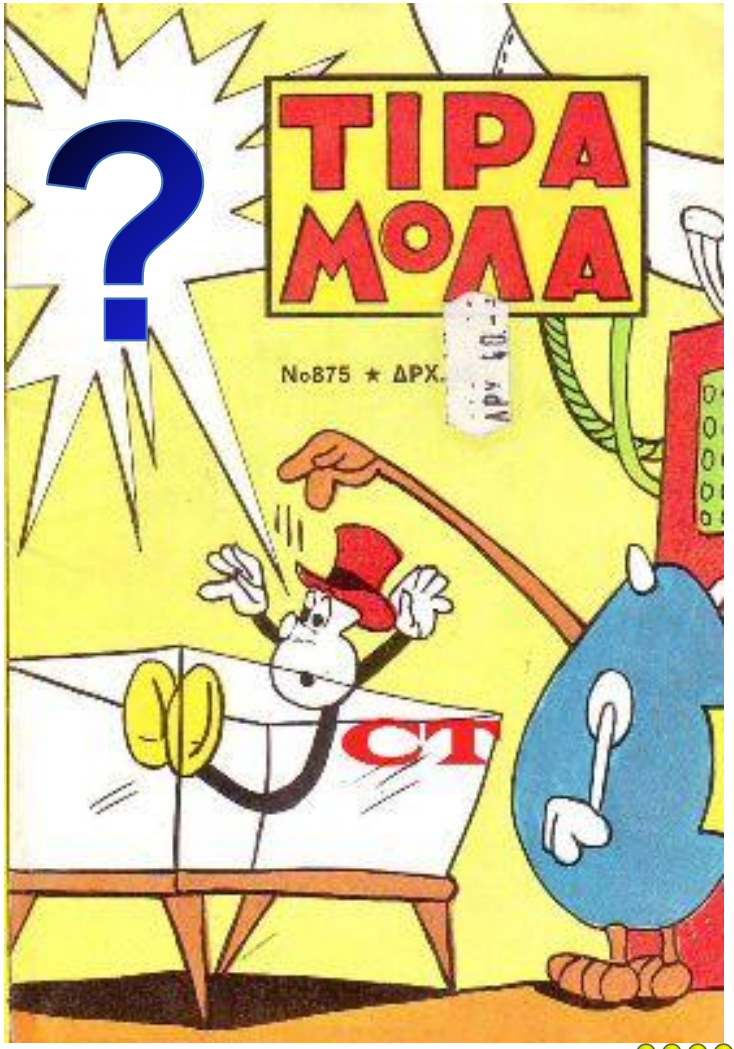
Conclusions – best practices (1)

- Choose the right DB for your application/workload (when in doubt, go with the one you're familiar with)
- HBase is a better all-rounder; Cassandra is handicapped by slow read performance and absence of shared FS.

Conclusions – best practices (2)

- **TIRAMOLA** is robust and in principle can be expanded for any kind of NoSQL DB or application by writing ~100 lines in Python.
- Building PaaS over IaaS is critical for the Cloud – most users won't have the knowledge, inclination, time or money to do it themselves, but need PaaS tools (in our example, elastic NoSQL databases).

Questions



National Technical University of Athens
CSLab