CamCube

Rethinking the Data Center Cluster

joint work with
Ant Rowstron, Austin Donnelly, and Greg O’Shea (MSR Cambridge)
Hussam Abu-Libdeh, Simon Schubert (Interns)
A Cloudy World
• Data Centers
  – Large group of networked servers
  – Typically built from commodity hardware
• Scale-out vs. scale-up
  – Many low-cost PCs rather than few expensive high-end servers
  – Economies of scale
• Custom software stack
  – Microsoft: Autopilot, Dryad
  – Google: GFS, BigTable, MapReduce
  – Yahoo: Hadoop, HDFS
  – Amazon: Dynamo, Astrolabe
  – Facebook: Cassandra, Scribe, HBase
Building large-scale services is hard!

- Mismatch between abstraction & reality
- How programmers see the network

- Key-based
- Overlay networks
- Explicit structure
- Symmetry of role
- ...
How the network really looks like...
How the network really looks like...

- ~12 racks / pod
- ~20-40 servers / rack
- 10 Gbps
How the network really looks like...

- **Oversubscription**
- **TCP Incast**

- ~12 racks / pod
- 10 Gbps
- Queue build-up
- ~ 20-40 servers / rack
- Fate-sharing

Paolo Costa
CamCube - Rethinking the Data Center Cluster
Data center services are distributed but they have no control over the network

- Must *infer* network properties (e.g., locality)
- Must use inefficient overlay networks
- No bandwidth guarantee

~ 20-40 servers / rack
• **Top-down (applications)**
  
  - *Custom routing?*
    - Overlays
  
  - *Priority traffic?*
    - Use multiple TCP flows
  
  - *Locality?*
    - Use traceroute and ping
  
  - *TCP Incast?*
    - Add random jitter

• **Bottom-up (network)**
  
  - *More bandwidth?*
    - Use richer topologies (e.g., fat-trees)
  
  - *Fate-sharing?*
    - Use prediction models to estimate flow length / traffic patterns
  
  - *TCP Incast?*
    - Use ECN in routers
Fundamental Issue

• The network is still a black box
  - Good for a multi-owned, heterogeneous Internet
  - Limiting for a single-owned, homogenous data center

• DCs are not mini-Internets
  - single owner / administration domain
  - we know (and define) the topology
  - low hardware/software diversity
  - no malicious or selfish node

• Is TCP/IP still appropriate?
The CamCube Goal

- “A data center from the perspective of a distributed systems builder”
- What happens if you integrate network & compute?
The High-level Story

- Service composition
- Link and key-based
- Architecture
- API
- Topology

Paolo Costa
CamCube - Rethinking the Data Center Cluster
Why A 3D Torus?

• No distinction between overlay and underlying network
• Nodes have \((x, y, z)\) coordinate
  – defines key-space
• Simple one hop API to send/receive packets
  – all other functionality implemented as services
Why A 3D Torus?

- No distinction between overlay and underlying network
- Nodes have \((x, y, z)\) coordinate – defines key-space
- Simple one hop API to send/receive packets – all other functionality implemented as services
- Key coordinates are re-mapped in case of failures – enable a KBR-like API
Packets move from service to service and server to server

Services can intercept and process packets along the way.

Keys are re-mapped in case of failures.

Server resources (RAM, Disks, CPUs) are accessible.

Fetch mail cost:a:1234
Packets move from service to service and server to server.

Keys are re-mapped in case of failures.

Server resources (RAM, Disks, CPUs) are accessible.

Example:

HotMail

hash(costa:1234) = (2,0,0)

(0,0,0)

(1,0,0)

(2,0,0)

Extract:

• Packets move from service to service and server to server
• Packets move from service to service and server to server
• Packets move from service to service and server to server
• Packets move from service to service and server to server
- Packets move from service to service and server to server

hash(costa:1234) = (2,0,0)

Route to (2,0,0)

Key-value store

Caching

Key-based routing

Key-value store

Caching

Key-based routing

Key-value store

Caching

Key-based routing

• Packets move from service to service and server to server
Packets move from service to service and server to server.

Services can intercept and process packets along the way.

Keys are remapped in case of failures.

Server resources (RAM, Disks, CPUs) are accessible to all services.

Example:

Fetch mail

costa:1234

Apps

Services

HotMail

CamCube - Rethinking the Data Center Cluster

Paolo Costa
• Packets move from service to service and server to server
• Packets move from service to service and server to server
• Packets move from service to service and server to server
• Services can intercept and process packets along the way
• Packets move from service to service and server to server
• Services can intercept and process packets along the way
• Packets move from service to service and server to server
• Services can intercept and process packets along the way
• Packets move from service to service and server to server
• Services can intercept and process packets along the way
• Packets move from service to service and server to server
• Services can intercept and process packets along the way
• Keys are re-mapped in case of failures
• Packets move from service to service and server to server
• Services can intercept and process packets along the way
• Keys are re-mapped in case of failures
• Packets move from service to service and server to server
• Services can intercept and process packets along the way
• Keys are re-mapped in case of failures
Packets move from service to service and server to server
Services can intercept and process packets along the way
Keys are re-mapped in case of failures
• Packets move from service to service and server to server
• Services can intercept and process packets along the way
• Keys are re-mapped in case of failures
• Server resources (RAM, Disks, CPUs) are accessible to all services
• 27-server (3x3x3) prototype
  – quad-core Intel Xeon 5520 2.27 Ghz
  – 12GB RAM
  – 6 Intel PRO/1000 PT 1 Gbps ports
  – Runtime & services implemented in C#
    • Multicast, key-value store, MapReduce, graph processing engine, ...

• Experiment
  – all servers saturating 6 links
  – 11.94 Gbps (9K pkts) using < 22% CPU
• **MapReduce**
  
  – originally developed by Google
  
  – open-source version (Hadoop)
    used by several companies
    
    • Yahoo, Facebook, Twitter, Amazon, ...

  – many applications
    
    • data mining, search, indexing...

• **Simple abstraction**

  – **map**: apply a function that generates (key, value) pairs

  – **reduce**: combine intermediate results
MapReduce from 10,000 ft

Paolo Costa
CamCube - Rethinking the Data Center Cluster
MapReduce today

- Results stored on a single node
- Can handle all reduce functions
- Bottleneck (network and CPU)

Paolo Costa
CamCube - Rethinking the Data Center Cluster
CPU & network load distributed

- $N^2$ flows: needs full bisection bw
- Results are split across R servers
- Not always applicable (e.g., max)
MapReduce conceptually
Use hierarchical aggregation!
MapReduce conceptually
Intermediate results can be aggregated at each hop
Camdoop: MapReduce in CamCube
Camdoop: MapReduce in CamCube
Camdoop: MapReduce

- Traffic reduced at each hop
- No need for full bisection bw
- Final results stored on a single node

In-network aggregation
• Camdoop builds 1 tree
  – failure resilient
Camdoop builds 1 tree of 6 disjoint trees
- failure resilient
- (up to) 6 Gbps throughput / server
Evaluation: Wordcount

- 27-server testbed
- 27 GB input size
  - 1GB / server
- Output size/ intermediate size ($S$)
  - $S=1/N \approx 0$ (full aggregation)
  - $S=1$ (no aggregation)
- Two configurations:
  - All-to-one
  - All-to-all
All-to-one

Worse

Better

Full aggregation
No aggregation

Time (s) logscale

Output size / intermediate size (S)

Switch
Camdoop (no agg)
Camdoop
Switch 1 Gbps (bound)
Impact of aggregation

Performance independent of \( S \)

Impact of higher bandwidth

Better aggregation ratio

Facebook reported aggregation ratio

Impact of higher bandwidth
All-to-one (small data size)

Worse

Better

Paolo Costa
CamCube - Rethinking the Data Center Cluster
In-network aggregation beneficial also for low-latency services.
Conclusions

• Building large-scale services is hard
  – mismatch between physical and logical topology
  – the network is a black box

• CamCube
  – designed from the perspective of developers
  – explicit topology and key-based abstraction

• Benefits
  ✓ simplified development
  ✓ more efficient applications and better fault-tolerance

