A Framework for an In-depth Comparison of Scale-up and Scale-out Systems

Michael Sevilla

Ike Nassi, Kleoni Ioannidou, Scott Brandt, Carlos Maltzahn

University of California, Santa Cruz

October 25, 2013
Scaling

Q: What do we do when there is too much data?

A: Scale the system

- **out**
  - ++ nodes to the system
  - → modify applications

- **up**
  - ++ resources to a single node
  - → modify the system
Scaling

Q: What do we do when there is too much data?

A: Scale the system

▶ out
  ++ nodes to the system
  → modify applications

▶ up
  ++ resources to a single node
  → modify the system

Q: Which is better?
Goal

Framework for comparing:

Why re-examine scale-up?

- new technology
- simplicity
- legacy applications
Goal

Framework for comparing:

Limit study to MapReduce
- standard for big data analytics
- goal is to make fair comparisons
Challenges - How do we:

Q: compare algorithms?
Q: compare hardware?
Q: account for properties provided by scale-out by default?

By design, scale-out provides:

- parallelism by automatically distributing load
- fault tolerance by rescheduling computation
- scalable storage with a distributed file system
- portability; Hadoop applications can run on any cluster
- availability because it can continually service clients
- scalability
Challenges - How do we:

Q: *compare algorithms?*
Q: *compare hardware?*
Q: *account for properties provided by scale-out by default?*

By design, scale-out provides:

- parallelism by automatically distributing load
- fault tolerance by rescheduling computation
- scalable storage with a distributed file system
- portability; Hadoop applications can run on any cluster
- availability because it can continually service clients
- scalability

These properties may or may not affect performance...

...but they can’t be ignored!
Challenges - How do we:

Q: compare algorithms?
Q: compare hardware?
Q: account for properties provided by scale-out by default?

By design, scale-out provides:

- parallelism by automatically distributing load
- fault tolerance by rescheduling computation
- scalable storage with a distributed file system
- portability; Hadoop applications can run on any cluster
- availability because it can continually service clients
- scalability

These properties may or may not affect performance...

...but they can’t be ignored!
Contributions

1. Comparison framework for scale-out/up
2. Implement scale-out properties on scale-up
   - Parallelism limited by new job phases
   - Fault tolerance can make scale-up slower than scale-out
   - Scalable storage may be the ultimate bottleneck

We show:
- must consider properties when comparing scale-out/up
- fundamental limitations of each architecture
Related Work

Scale-up vs. Scale-out Hadoop: Time to Rethink?

ACM Symposium on Cloud Computing ’13 [1, 2]

- 10 “typical” jobs
- for today’s jobs, scale-up server > scale-out cluster
Methodology: comparison parameters

**input**
- ►

**software**
- ►
- ►
- ►

**hardware**
- ►
Methodology: comparison parameters

input
- workload, input size

software

hardware

same workload, scale data

同一规模的工作负载，扩展数据

Scale-out vs. Scale-up

Introduction
- challenges
- contributions

Methodology
- parameters
- framework

Analysis
- initial results
- scale-out properties
- parallelism

Conclusion
Methodology: comparison parameters

input
▶ workload, input size

software
▶ problem

hardware

→ same workload, scale data

software
▶ problem

→ word count, sort

hardware

→

Methodology: comparison parameters

scale-up vs. scale-out

Introduction
challenges
contributions
Methodology
parameters
framework
Analysis
initial results
scale-out properties
parallelism
Conclusion
Methodology: comparison parameters

input
► workload, input size

software
► problem
► algorithm

hardware

→ same workload, scale data

→ word count, sort

→ methodology, functionality

→

scale-up vs. scale-out
Michael Sevilla
Introduction challenges contributions
Methodology parameters framework
Analysis initial results scale-out properties parallelism
Conclusion
Methodology: comparison parameters

input
- workload, input size

software
- problem
- algorithm
- scale-out properties

→ same workload, scale data

software

→ word count, sort
→ methodology, functionality
→ implementations

Conclusion
Methodology: comparison parameters

input
► workload, input size

software
► problem
► algorithm
► scale-out properties

hardware
► processors, memory

→ same workload, scale data

→ word count, sort
→ methodology, functionality
→ implementations

→ ≡ compute contexts
Experimental Setup

Scale-out
- 32 nodes, 2-dual core processors, 8GB RAM

Scale-up
- 1 node, 2-quad core processors (HT), 256GB RAM
  = 32 compute contexts, 256GB RAM
* node ∈ scale-out gets the same work as a thread ∈ scale-up
Scale-up can Perform Better than Scale-out

![Graph showing word count execution times for scale-out and parallel scale-up. The graph plots Wall clock time (seconds) on the y-axis and Input size (GB) on the x-axis. The graph shows that scale-out has a steeper increase in wall clock time compared to parallel scale-up, indicating that scale-up can perform better at large input sizes.](image-url)
... but achieving scale-out properties changes the story!

▶ Other properties must be considered in comparison
Achieving Scale-out Properties on Scale-up

Phoenix: MapReduce runtime for multicore systems
→ parallelism, port for methodology

Distributed MultiThreaded Checkpointing (DMTCP)
→ fault tolerance

Hadoop Distributed File System (HDFS)
→ scalable storage
Achieving Scale-out Properties on Scale-up

Properties must be considered when comparing scale-out/up

Word count execution times

- Scale-out
- Sequential scale-up
- Parallel scale-up

Wall clock time (seconds)

Input size (GB)
Achieving Scale-out Properties on Scale-up

Properties must be considered when comparing scale-out/up
- fault tolerance can make scale-up slower than scale-out
Achieving Scale-out Properties on Scale-up

Properties must be considered when comparing scale-out/up

- fault tolerance can make scale-up slower than scale-out
- scalable storage may be the ultimate bottleneck
Achieving Scale-out Properties on Scale-up

Properties must be considered when comparing scale-out/up
- fault tolerance can make scale-up slower than scale-out
- scalable storage may be the ultimate bottleneck
- parallelism limited by new job phases

Properties must be considered when comparing scale-out/up
- fault tolerance can make scale-up slower than scale-out
- scalable storage may be the ultimate bottleneck
- parallelism limited by new job phases
Achieving Parallelism

Wall clock time (seconds)

Input size (GB)

Word count

Sort

Wall clock time (seconds)

Input size (GB)

Sort execution times

scale-out
sequential s-up
parallel s-up

scale-out
sequential s-up
parallel s-up
Achieving Parallelism

Why is sort slower?!
Parallelism is limited by new job phases
  ▶ read: move data from disk into memory
  ▶ merge: sort the data
Sort is slower on scale-up because
  1. new job phases
  2. more key-value pairs
Conclusion

Compare scaling architectures (scale-up/out)

▶ comparison framework
  → encompasses \{input, software, hardware\} parameters
▶ implement scale-out properties on scale-up

We show:

▶ must consider properties when comparing scale-out/up
▶ fundamental limitations of each architecture
Questions?

Special thanks to our anonymous reviewers for their helpful comments and suggestions. Also, thanks to Joe Buck and Noah Watkins for explaining to me what my results mean. M.

Scale-up vs Scale-out for Hadoop: Time to Rethink?

Nobody Ever Got Fired for using Hadoop on a Cluster.
Sequential applications

Word count execution times

Sort execution times
Achieving Parallelism…

- Using OpenMP

Parallelism still limited by new job phases
Achieving Fault Tolerance...

- ↓ performance, parallelism, scalable storage, availability, portability

Fault tolerance can make scale-up less appealing
Achieving Fault Tolerance...

- ↓ performance, parallelism, scalable storage, availability, portability

![Checkpoint intervals graph](image)

- Fault tolerance can make scale-up less appealing
Achieving Scalable Storage...

- ↓ performance
- ↑ parallelism, fault tol., portability

Scalable storage may be the ultimate bottleneck