Comparing Scale-up and Scale-out...

... an Empirical Study... ?

Michael Sevilla

University of California, Santa Cruz

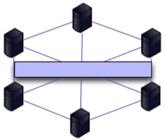
March 18, 2013

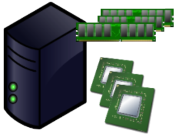
Scaling

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





Q: *Which is better?*

Current Trend

1. push towards scale-out (past)

2. difficulty of scale-out (present)

3. push towards scale-up (future?)

Current Trend

1. push towards scale-out (past)
   - hardware
     - non-linear scaling
   - cost
     - 1 expensive node vs. many commodity servers
   - interoperability
     - OSs not designed for ++ resources
     - Barrelfish, FOS, Corey, Cerberus [15, 19, 1, 16]
     - Linux scalability, LANL study [2, 3]

Result: MapReduce, Dryad [6, 9]

Current Trend

1. push towards scale-out (past)
   - hardware
   - cost
   - interoperability

2. difficulty of scale-out (present)
   - workload specific architectures
     - Pregel, Spark, S4 [10, 21, 12]
   - application optimization
     - concurrent programming
     - parallel databases
     - resource management
   - complexity, unpredictability
     - NFS death spirals
     - faulty network interfaces
     - 100% CPU utilization on the gateway
     - namode/t-trackers/tmp won't format/start/reset

Maintaining a cluster is not fun

11	Adam Crume	12.04	SSD, bad ram chip	edid (nomodeset)
12	Joe Buck	12.04	SSD (ssd looks flakey,cannot do an fdisk)	edid (nomodeset)
13	Joe Buck	12.04	SSD	edid (nomodeset)
14	Joe Buck	12.04	SSD	edid (nomodeset)
15	Adam Crume	12.04		can't boot from cd-rom (use usb?)
16	Joe Buck	12.04	SSD	edid (nomodeset)
17	Joe Buck	12.04	SSD	edid (nomodeset)
18	Noah	12.04	SSD	edid (nomodeset)
19	Joe Buck	12.04	SSD	edid (nomodeset)
20		12.04	SSD issues with 2 hard drives	edid (nomodeset)
21	Joe Buck	12.04	SSD	edid (nomodeset)
22	Joe Buck	12.04	SSD: looks okay, keep an eye on it	edid (nomodeset)
23	Joe Buck	12.04	SSD	edid (nomodeset)
24	Noah	12.04	SSD	edid (nomodeset)
25	Joe Buck	12.04	SSD	edid (nomodeset)
26	Joe Buck	12.04	SSD this looks okay. Keep an eye on it	edid (nomodeset)
27	Noah	12.04	SSD	edid (nomodeset)
28	Noah	12.04	SSD	edid (nomodeset)
29	Noah	12.04	SSD	edid (nomodeset)
30	Noah	12.04	SSD	edid (nomodeset)
31	Noah	12.04		edid (nomodeset)
32	Noah	12.04		
33	**down**		RAM was pulled to fix another host. Replaced RAM is in Joe's desk (at least 1 bad chip in the bunch)	
34	Joe Buck	12.04		
35	Joe Buck	12.04		edid (nomodeset)
36	No BIOS or POST			edid (nomodeset)
37	Joe Buck	12.04		edid (nomodeset)
38	Joe Buck	12.04		edid (nomodeset)
39	Noah	12.04		edid (nomodeset)
40	Joe Buck	12.04		
41	Noah	12.04		
42				
43			keep an eye on this node. Is not booting. Not sure why.	
44	Noah	12.04		
45	Michael Sevilla		issdm SSH key not set	
46	Noah	12.04	keep an eye on this node	
47	Joe Buck	12.04	keep an eye on this node	

Current Trend

1. push towards scale-out (past)
   - hardware
   - cost
   - interoperability

2. difficulty of scale-out (present)
   - workload specific architectures
   - application optimization
   - complexity, unpredictability

Current Trend

1. push towards scale-out (past)
   - hardware
   - cost
   - interoperability
2. difficulty of scale-out (present)
   - workload specific architectures
   - application optimization
   - complexity, unpredictability
3. push towards scale-up (future?)
   - simplicity
   - automization
   - evolve

Proposal: new scale-up vs. scale-out study

Proposal: new scale-up vs. scale-out study

Why? Because previous studies use:

1. narrow methodologies
   - distr. sys: # of nodes [15, 1, 16, 19, 13, 20, 17, 18, 2]
   - distr. sys: workload types [5, 21, 10]
   - single node: threads/cores [14, 7, 8, 9, 4]
2. out-dated systems
   - (8 $\times$ dual-core, 32GB RAM) vs. (14 nodes) [11]

Figure: The POWER5 p5 575 SMP server.

Proposal: new scale-up vs. scale-out study

Why? Because previous studies use:

1. narrow methodologies
2. out-dated systems

→ missing bottlenecks

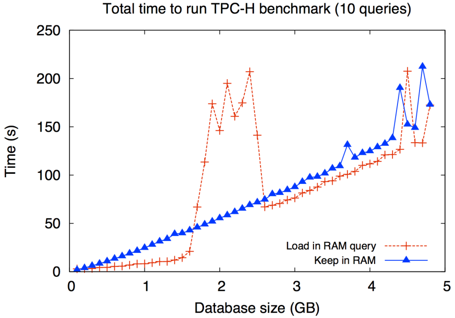
Proposal: new scale-up vs. scale-out study

Why? Because previous studies use:

1. narrow methodologies
2. out-dated systems

→ missing bottlenecks

- **Key Observation**: big data uses a lot of data



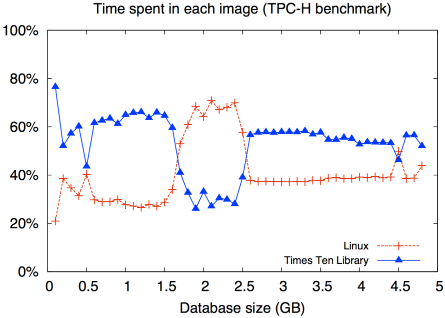
Proposal: new scale-up vs. scale-out study

Why? Because previous studies use:

1. narrow methodologies
2. out-dated systems

→ missing bottlenecks

- **Key Observation**: big data uses a lot of data



Proposal: new scale-up vs. scale-out study

Why? Because previous studies use:

1. narrow methodologies
2. out-dated systems

→ missing bottlenecks

- **Hypothesis**: there will be new bottlenecks/slowdowns

Proposal: new scale-up vs. scale-out study

Why? Because previous studies use:

1. narrow methodologies
2. out-dated systems

→ missing bottlenecks

- Methodology: vary machine configs + data

Methodology: vary machine configs + data

Long-term goal: construct performance grid

	M_1	M_2	M_3	$\dots$	M_n
A_1	p_{11}	p_{12}	p_{13}	$\dots$	
A_2	p_{21}	p_{22}	p_{23}		
$A_1{}'$	$p_{11}{}'$	$p_{12}{}'$	$p_{13}{}'$		
$A_2{}'$	$p_{21}{}'$	$p_{22}{}'$	$p_{23}{}'$		
$\vdots$	$\vdots$				
A_m					

This will help us:

1. create a cost function
2. identify the differences between scaling out and up

Methodology: vary machine configs + data

Short-term goal: small experiment comparing scale-out/up

```
foreach application
    while(!stressed)
        execute()
        measure_performance()
        ++data
```

Problem: how do we select applications?

- representative and feasible

Methodology: vary machine configs + data

Short-term goal: small experiment comparing scale-out/up

```
foreach application
    while(!stressed)
        execute()
        measure_performance()
        ++data
```

Problem: how do we port applications?

- functionality or methodology?
- fair and feasible

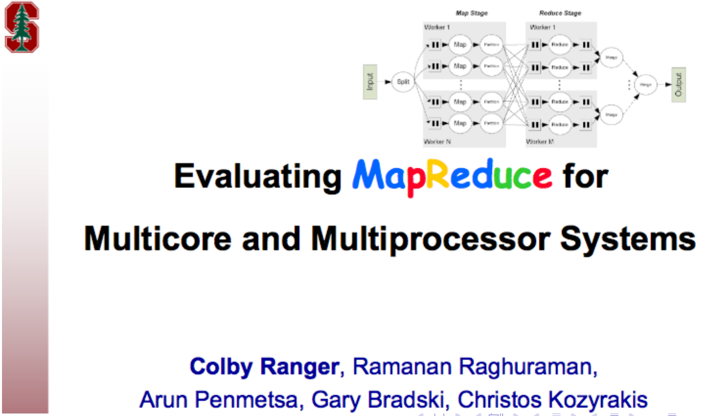
Implementation

Select apps.: existing distr. sys. benchmark (HiBench) [8]

- word count, sort, Terasort, PageRank, Nutch

Port apps.: Phoenix API/runtime

- MapReduce → multi-{core, processor}



Phoenix vs. MapReduce

	MapReduce	Phoenix
work distr.	master node worker nodes	parent process threads $\in$ core
communication	network i-keys $\in$ HDFS	shared-memory i-keys $\in$ L1 cache
fault tolerance	heartbeat remote re-exec.	timeout local re-exec.
combiner	$\in$ node after map	$\in$ thread after map

This makes our comparison:

- ✓ fair
- ✓ feasible
- ✓ representative

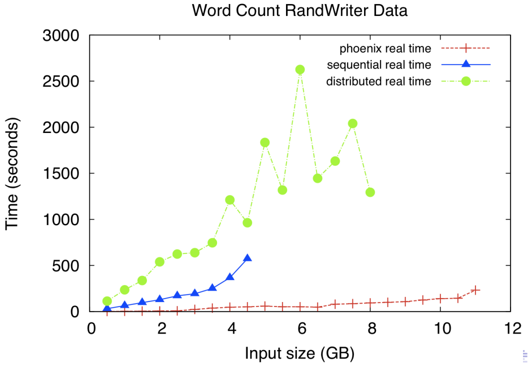
Porting progress

scale-out Hadoop (3 nodes)	scale-up ≡ methodology	scale-up ≡ functionality
✓ WordCount.java	✓ wc.cpp	✓ wc-seq.cpp
✓ Sort.java	✓ sort.cpp	✓ sort-seq.cpp.
✓ TeraSort.java	✗ tsort.cpp	✗ tsort-seq.cpp
✓ Hama	✗ pg_rank.cpp	✗ pg_rank-seq.cpp
✓ SolrIndex.java	✓ index.cpp	✓ index-seq.cpp

Initial Results: word count

	Data	Time	Error → Event	
`wc.cpp`	11.5 GB	232.82 secs	cpu throttled → `int_idle()`	+10%
`wc-seq.cpp`	4.5 GB	572.75 secs	bad allocation → `scan_swap()`	+8%

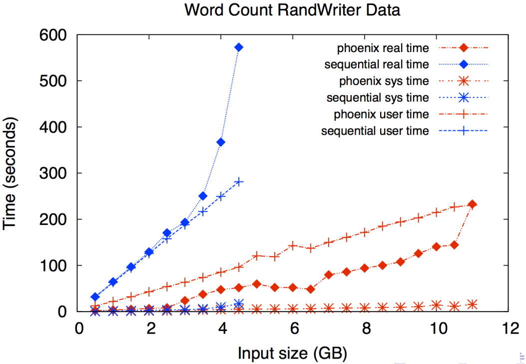
- scale-out vs. scale-up



Initial Results: word count

	Data	Time	Error → Event	
`wc.cpp`	11.5 GB	232.82 secs	cpu throttled → `int_idle()`	+10%
`wc-seq.cpp`	4.5 GB	572.75 secs	bad allocation → `scan_swap()`	+8%

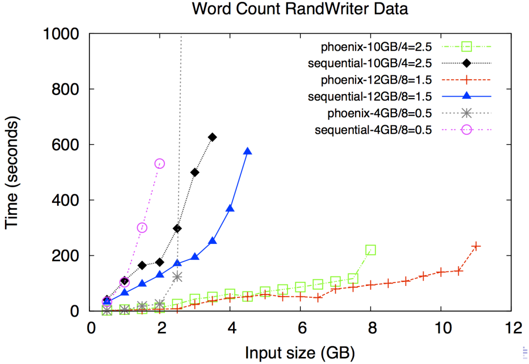
- scale-up timing breakdown



Initial Results: word count

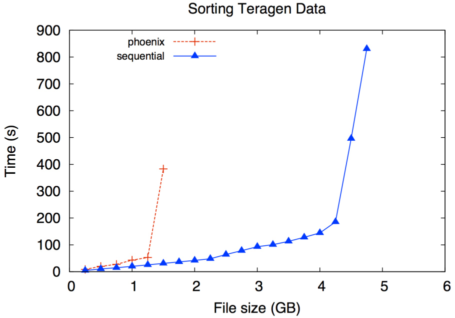
	Data	Time	Error → Event	
`wc.cpp`	11.5 GB	232.82 secs	cpu throttled → `int_idle()`	+10%
`wc-seq.cpp`	4.5 GB	572.75 secs	bad allocation → `scan_swap()`	+8%

- scale-up $\frac{\text{mem}}{\text{core}}$ ratio breakdown



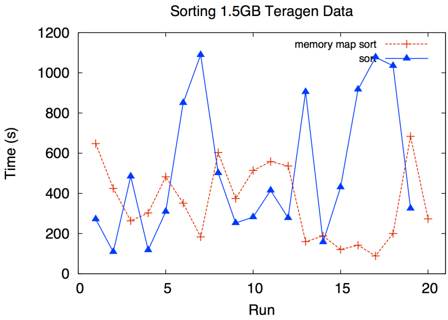
Initial Results: sort

	Data	Time	Error → Event	
`sort.cpp`	1.5	208.11	OOM; kill →	(+?%)
`sort-seq.cpp`	4.75	830.46	OOM; kill → `scan_swap()`	(+20%)



Initial Results: sort

	Data	Time	Error → Event	
`sort.cpp`	1.5	208.11	OOM; kill →	(+?%)
`sort-seq.cpp`	4.75	830.46	OOM; kill → `scan_swap()`	(+20%)



Conclusion

Lays the groundwork for scale-up vs. out study

- choose applications
- port applications
- methodology

The plan:

Spring Quarter

- port applications
- profile/take measurements
- write Masters Thesis

Summer

- intern @ TidalScale
- hands-on experience

Fall Quarter

- document summer expereince
- write paper?

Experience with Phoenix

```
class WordsMR : public MapReduceSort<...>{
    void map(data_type s, map_container out){
        wc_word word = { s.data+start };
        emit_intermediate(out, word, 1);
    }
    ...
    int split(wc_string& out){
        out.data = data + splitter_pos;
        out.len = end - splitter_pos;
    }
    ...
    bool sort(keyval a, keyval b){
        return a.val < b.val || ...;
    }
}
...
mapReduce.run()
```

References I

S. Boyd-Wickizer, H. Chen, R. Chen, Y. Mao, F. Kaashoek, R. Morris, A. Pesterev, L. Stein, M. Wu, Y. Dai, Y. Zhang, and Z. Zhang.
Corey: an operating system for many cores.
In *Proceedings of the 8th USENIX conference on Operating systems design and implementation*, OSDI'08, pages 43–57, Berkeley, CA, USA, 2008. USENIX Association.

S. Boyd-Wickizer, A. T. Clements, Y. Mao, A. Pesterev, M. F. Kaashoek, R. Morris, and N. Zeldovich.
An analysis of linux scalability to many cores.
In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI '10)*, Vancouver, Canada, October 2010.

S. S. P. G. Bridges and A. B. Maccabe.
A framework for analyzing linux system overheads on hpc applications.
In *Proceedings of the 2005 Los Alamos Computer Science Institute (LACSI '05)*, page 17, 2005.

F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber.
Bigtable: a distributed storage system for structured data.
In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, OSDI '06, pages 15–15, Berkeley, CA, USA, 2006. USENIX Association.

Y. Chen, A. Ganapathi, R. Griffith, and R. Katz.
The case for evaluating mapreduce performance using workload suites.
In *Proceedings of the 2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, MASCOTS '11, pages 390–399, Washington, DC, USA, 2011. IEEE Computer Society.

References II

J. Dean and S. Ghemawat.
Mapreduce: simplified data processing on large clusters.
In *Proceedings of the 6th conference on Symposium on Operarting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

Z. Fadika, M. Govindaraju, S. R. Canon, and L. Ramakrishnan.
Evaluating hadoop for data-intensive scientific operations.
In R. Chang, editor, *IEEE CLOUD*, pages 67–74. IEEE, 2012.

S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang.
The hibench benchmark suite: Characterization of the mapreduce-based data analysis.
In *ICDE Workshops*, pages 41–51, 2010.

M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly.
Dryad: distributed data-parallel programs from sequential building blocks.
In *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, EuroSys '07, pages 59–72, New York, NY, USA, 2007. ACM.

G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski.
Pregel: a system for large-scale graph processing.
In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM.

M. Michael, J. Moreira, D. Shiloach, and R. Wisniewski.
Scale-up x scale-out: A case study using nutch/lucene.
In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8. IEEE, 2007.

References III

L. Neumeyer, B. Robbins, A. Nair, and A. Kesari.
S4: Distributed stream computing platform.
In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, ICDMW '10, pages 170–177, Washington, DC, USA, 2010. IEEE Computer Society.

C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis.
Evaluating mapreduce for multi-core and multiprocessor systems.
In *Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture*, HPCA '07, pages 13–24, Washington, DC, USA, 2007. IEEE Computer Society.

B. Schroeder and G. A. Gibson.
A large-scale study of failures in high-performance computing systems.
In *Proceedings of the International Conference on Dependable Systems and Networks*, DSN '06, pages 249–258, Washington, DC, USA, 2006. IEEE Computer Society.

A. Schpbach, S. Peter, A. Baumann, T. Roscoe, P. Barham, T. Harris, and R. Isaacs.
Embracing diversity in the barrelfish manycore operating system.
In *In Proceedings of the Workshop on Managed Many-Core Systems*, 2008.

X. Song, H. Chen, R. Chen, Y. Wang, and B. Zang.
A case for scaling applications to many-core with os clustering.
In *Proceedings of the sixth conference on Computer systems*, EuroSys '11, pages 61–76, New York, NY, USA, 2011. ACM.

J. Talbot, R. M. Yoo, and C. Kozyrakis.
Phoenix++: modular mapreduce for shared-memory systems.
In *Proceedings of the second international workshop on MapReduce and its applications*, MapReduce '11, pages 9–16, New York, NY, USA, 2011. ACM.

A. Talkington and K. Dixit.
Scaling-up or out.
International Business, 2002.

References IV

D. Wentzlaff and A. Agarwal.
Factored operating systems (fos): the case for a scalable operating system for multicores.
ACM SIGOPS Operating Systems Review, 43(2):76–85, 2009.

R. M. Yoo, A. Romano, and C. Kozyrakis.
Phoenix rebirth: Scalable mapreduce on a large-scale shared-memory system.
In *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, IISWC '09, pages 198–207, Washington, DC, USA, 2009. IEEE Computer Society.

M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica.
Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing.
In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, NSDI'12, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.