RELEASE

http://www.release-project.eu

Scalable Persistent Storage for Erlang
Theory and Practice

Amir Ghaffari

Natalia Chechina , Phil Trinder

University of Glasgow

Jon Meredith

basho

October 22, 2013

RELEASE

Outline

- RELEASE Project
- General principles of scalable DBMSs
- NoSQL DBMSs for Erlang
- Riak 1.1.1 Scalability in Practice
- Investigating the scalability of distributed Erlang
- Riak Elasticity
- Conclusion & Future work

RELEASE project

- RELEASE is an European project aiming to scale Erlang onto commodity architectures with 100,000 cores.

RELEASE project

The RELEASE consortium work at following levels:

- ➢ Virtual machine
- ➢ Language
    - ▪ scalable Computation model
    - ▪ Scalable In-memory data structures
    - ✓ Scalable Persistent data structures
- ➢ Infrastructure levels
- ➢ Profiling and refactoring tools

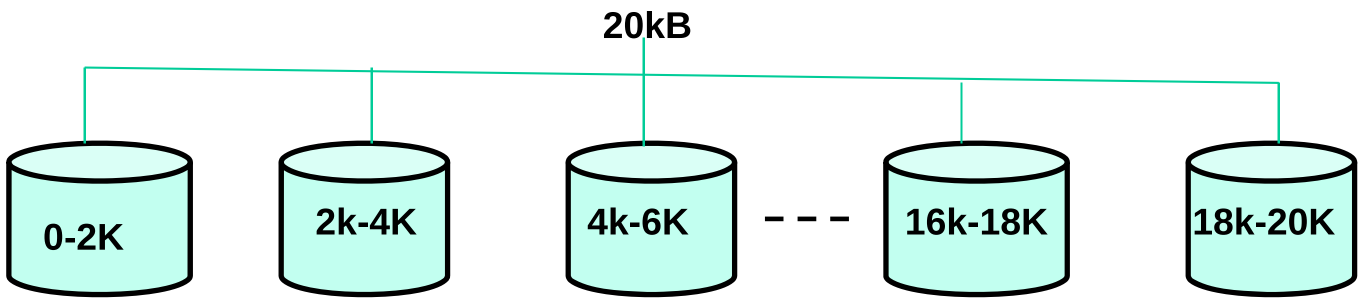
RELEASE

General principles of scalable DBMSs

Data Fragmentation

1. Decentralized model (e.g. P2P model)
2. Systematic load balancing (make life easier for developer)
3. Location transparency

e.g. 20k data is fragmented among 10 nodes



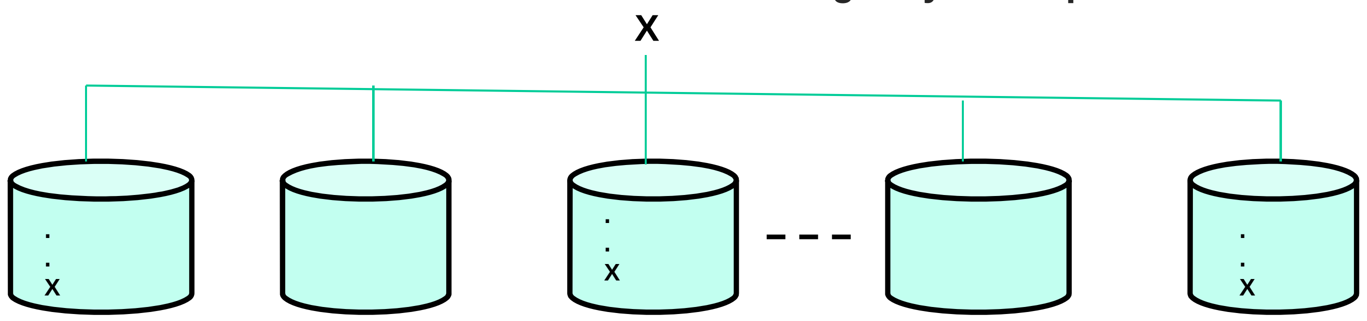
RELEASE

General principles of scalable DBMSs

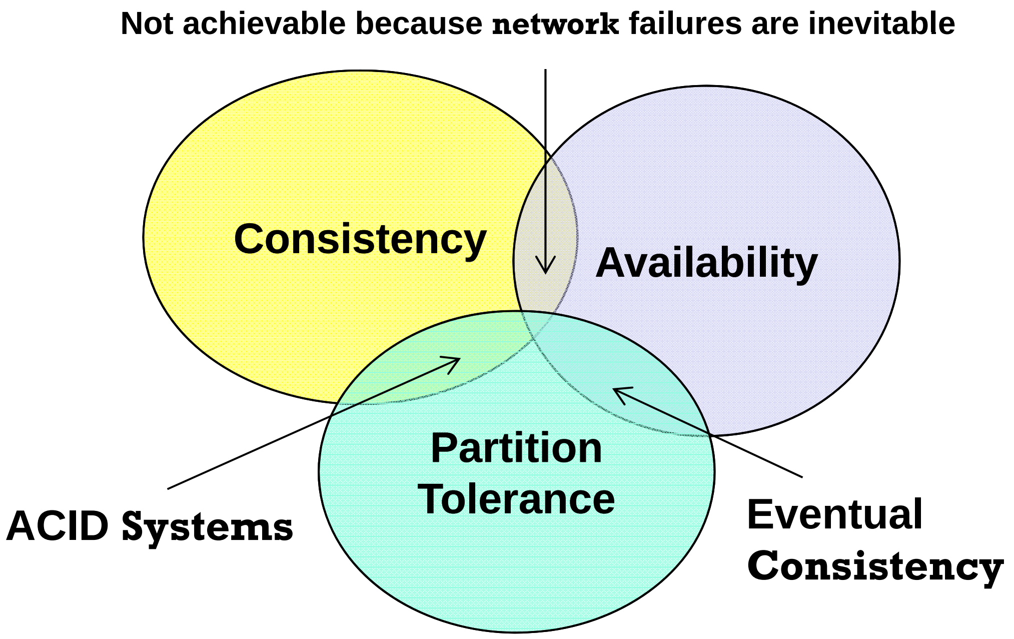
Replication

1. Decentralized model (e.g. P2P model)
2. Location transparency
3. Asynchronous replication (write is considered complete as soon as on node acknowledges it)

e.g. Key X is replicated on three nodes



General principles of scalable DBMSs



CAP theorem: cannot simultaneously guarantee:

- **Partition tolerance**: system continues to operate despite nodes can't talk to each other
- **Availability**: guarantee that every request receives a response
- **Consistency**: all nodes see the same data at the same time

Solution: Eventual consistency and reconciling conflicts via data versioning

ACID=Atomicity, Consistency, Isolation, Durability

RELEASE

NoSQL DBMSs for Erlang

	Mnesia	CouchDB	Riak	Cassandra
Fragmentation	•Explicit placement •Client-server •Automatic by using a hash function	•Explicit placement •Multi-server •Lounge is not part of each CouchDB node	•Implicit placement •Peer to peer •Automatic by using consistent hash technique	•Implicit placement •Peer to peer •Automatic by using consistent hash technique
Replication	•Explicit placement •Client-server •Asynchronous (Dirty operation)	•Explicit placement •Multi-server •Asynchronous	•Implicit placement •Peer to peer •Asynchronous	•Implicit placement •Peer to peer •Asynchronous
Partition Tolerant	•Strong consistency	•Eventual consistency •Multi-Version Concurrency Control for reconciliation	•Eventual consistency •Vector clocks for reconciliation	•Eventual consistency •Use timestamp to reconcile
Query Processing & Backend Storage	•The largest possible Mnesia table is 4Gb	•No limitation •Supports Map/Reduce Queries	•Bitcask has memory limitation •LevelDB has no limitation •Supports Map/Reduce queries	•No limitation •Supports Map/Reduce queries

RELEASE

Initial Evaluation Results

General Principles

Initial Evaluation

- Mnesia
- CouchDB
- Riak
- Cassendra

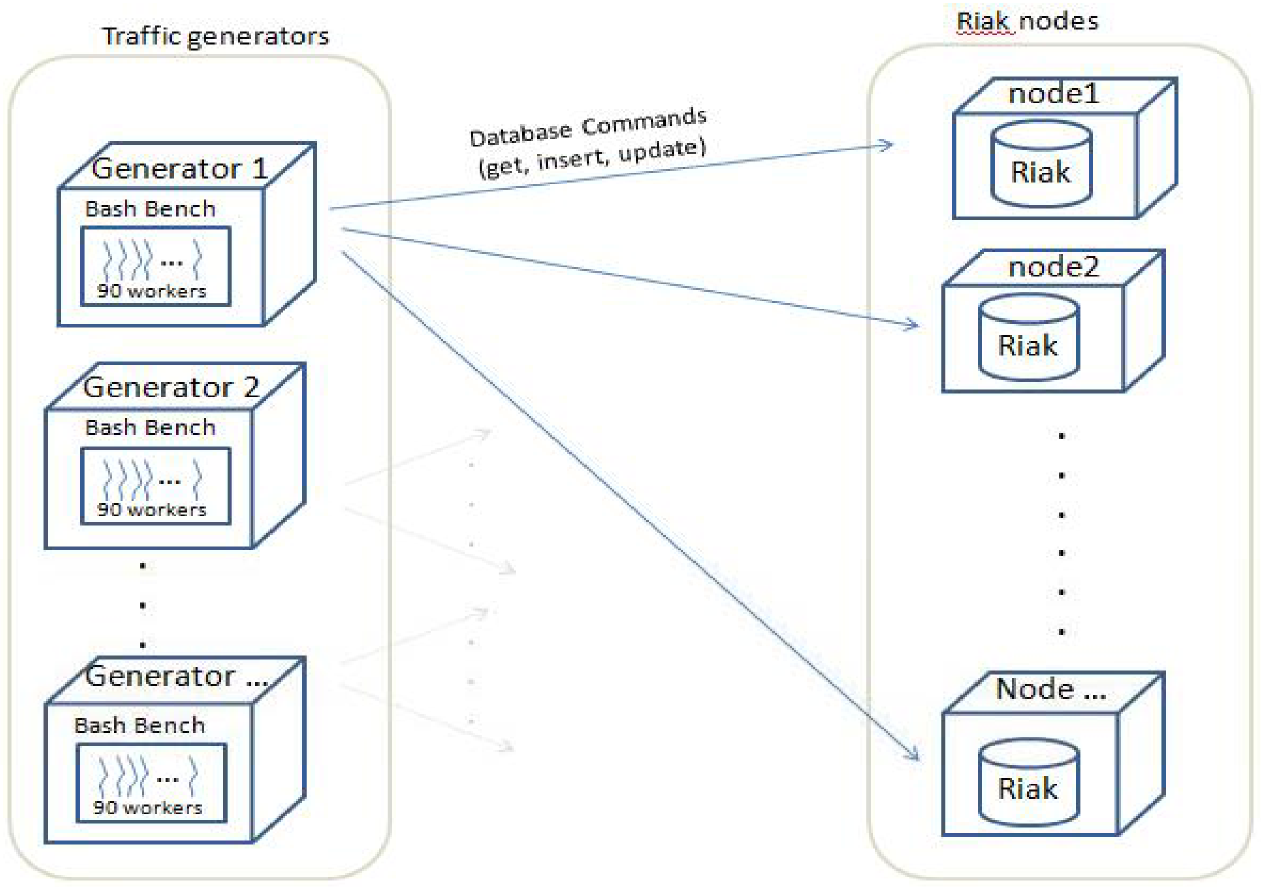
Scalable persistent storage for SD Erlang can be provided by Dynamo-style DBMSs such as *Riak,Cassandra*

Riak Scalability in Practice

- Basho Bench: a benchmarking tool for Riak
- We measure Basho Bench on 348-node Kalkyl cluster
- Scalability: How does adding more Riak nodes affect the throughput?
- There are two kinds of nodes in a cluster:
  - *Traffic generators*
  - *Riak nodes*

RELEASE

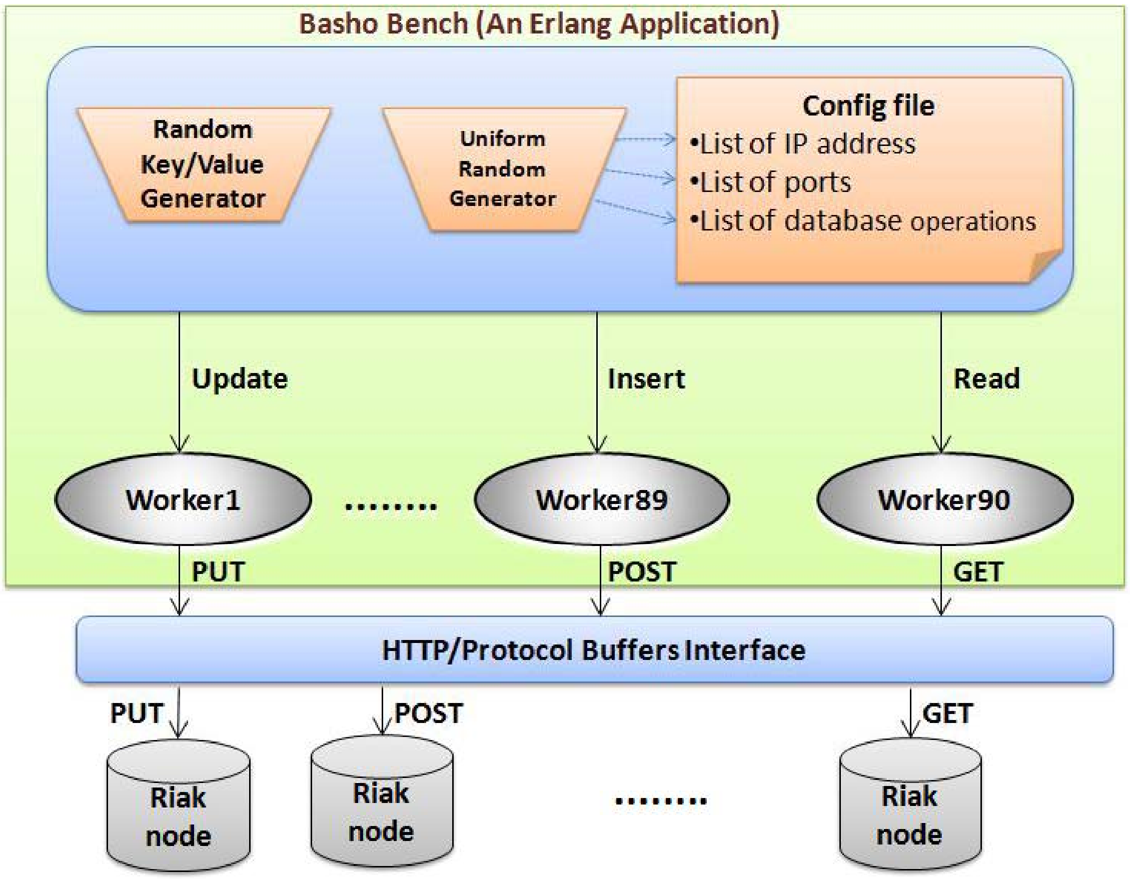
Node Organisation



Heuristic: one traffic generator per 3 Riak nodes

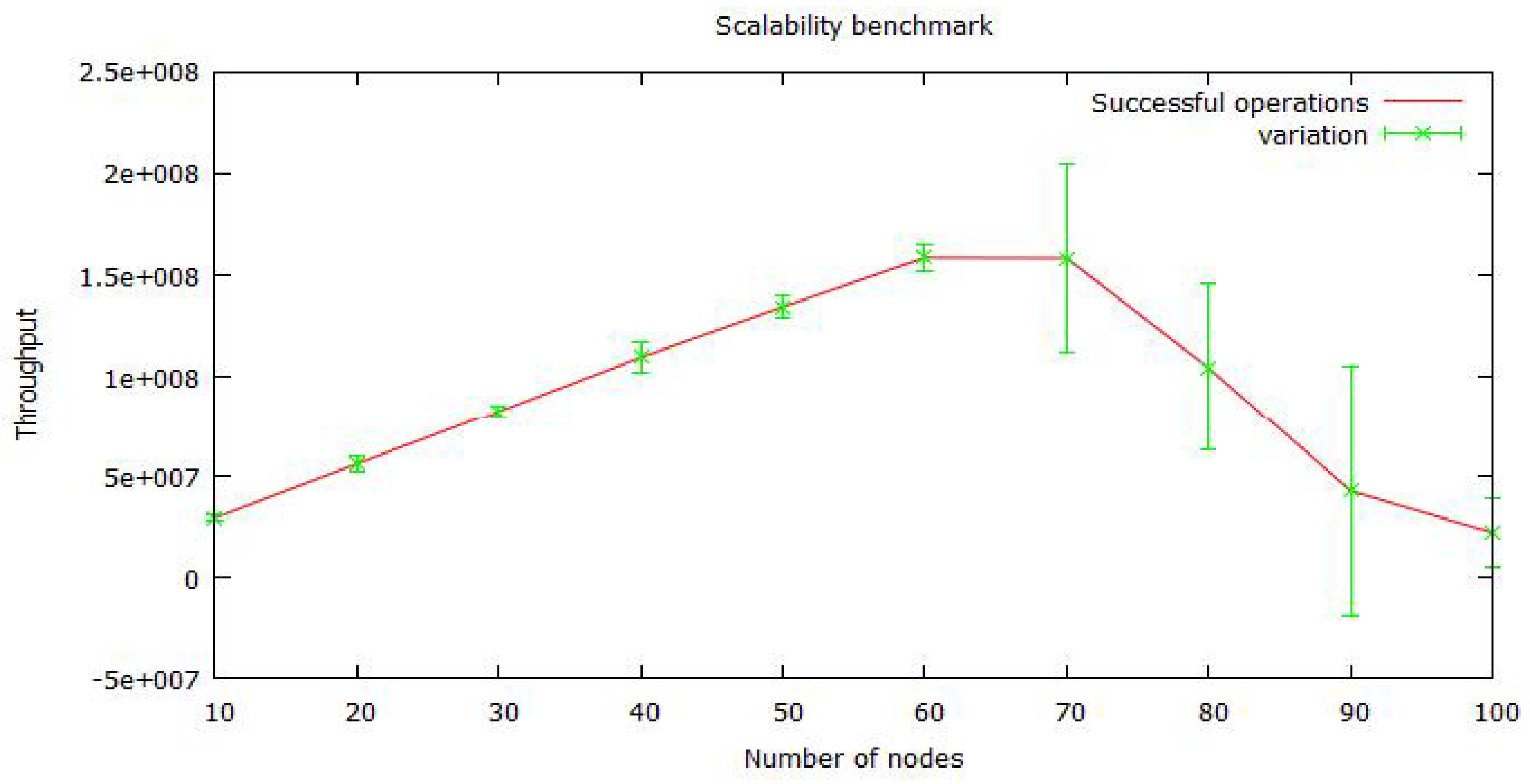
RELEASE

Traffic Generator



RELEASE

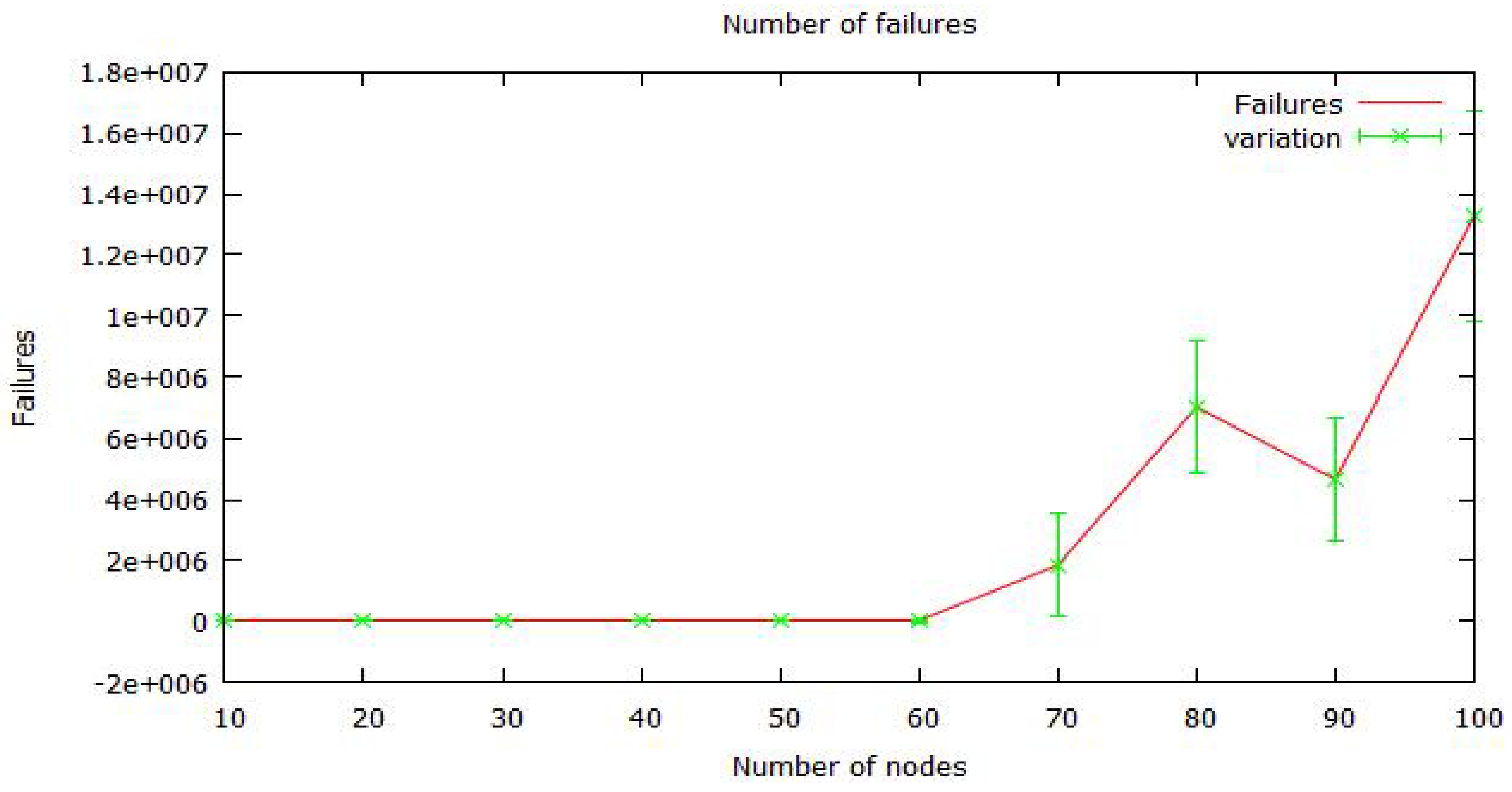
Riak 1.1.1 Scalability



Benchmark on 100-node cluster (800 cores)

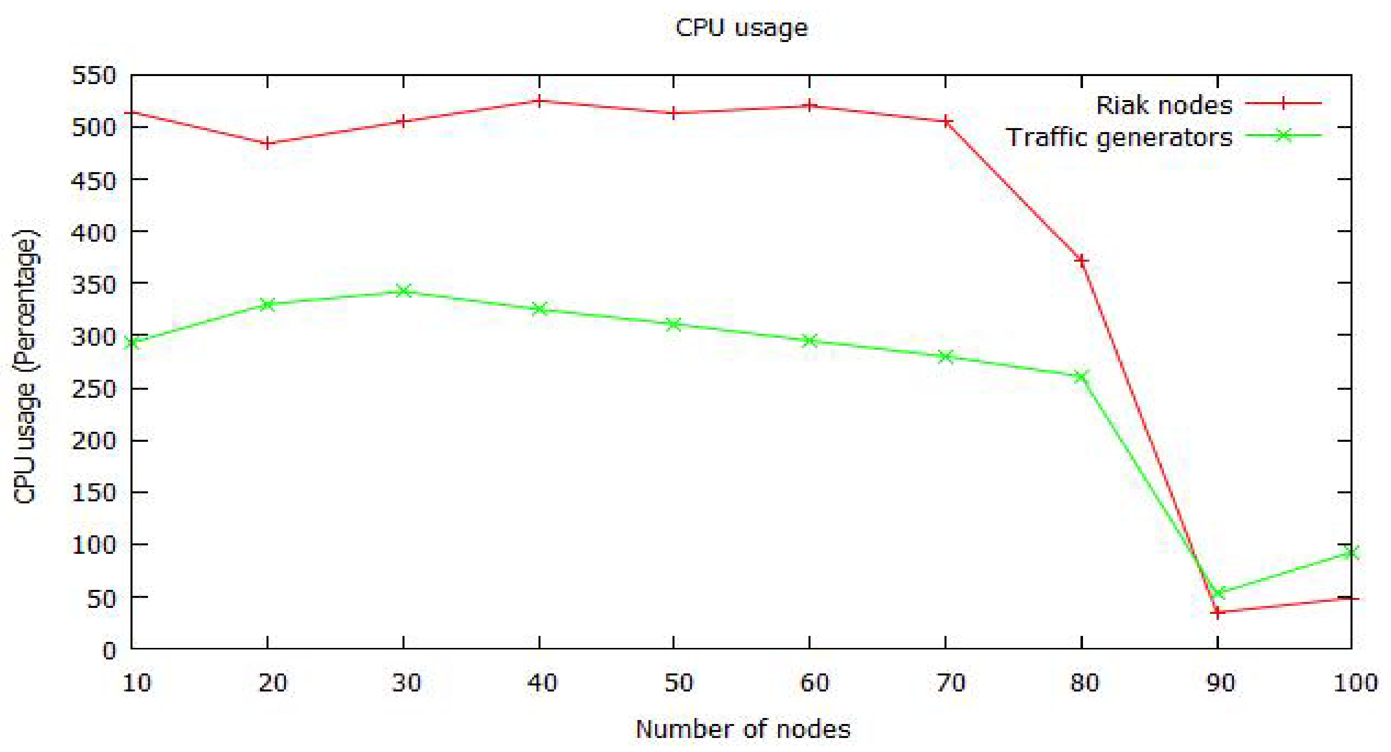
RELEASE

Failures



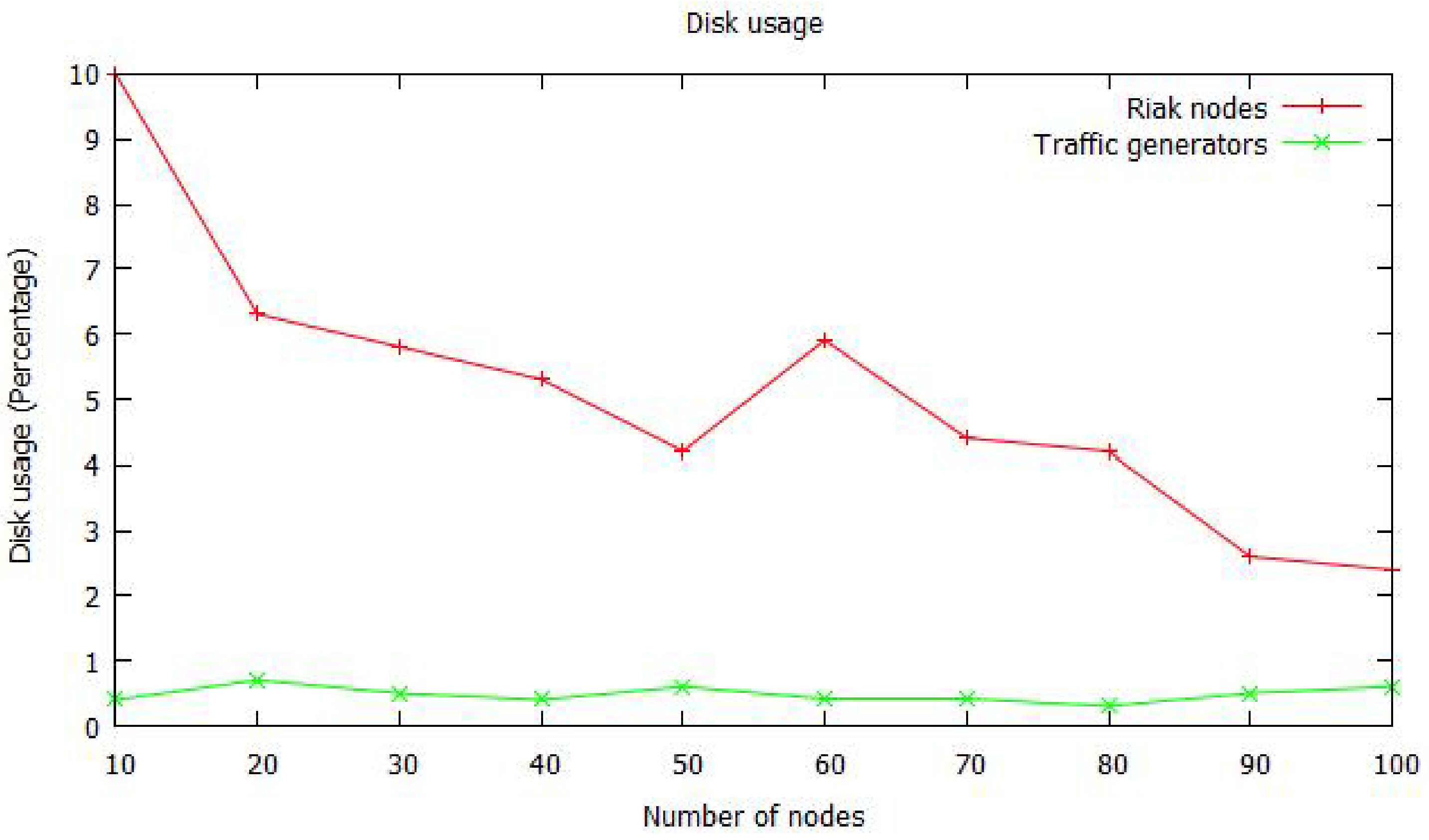
RELEASE

Profiling Resource Usage



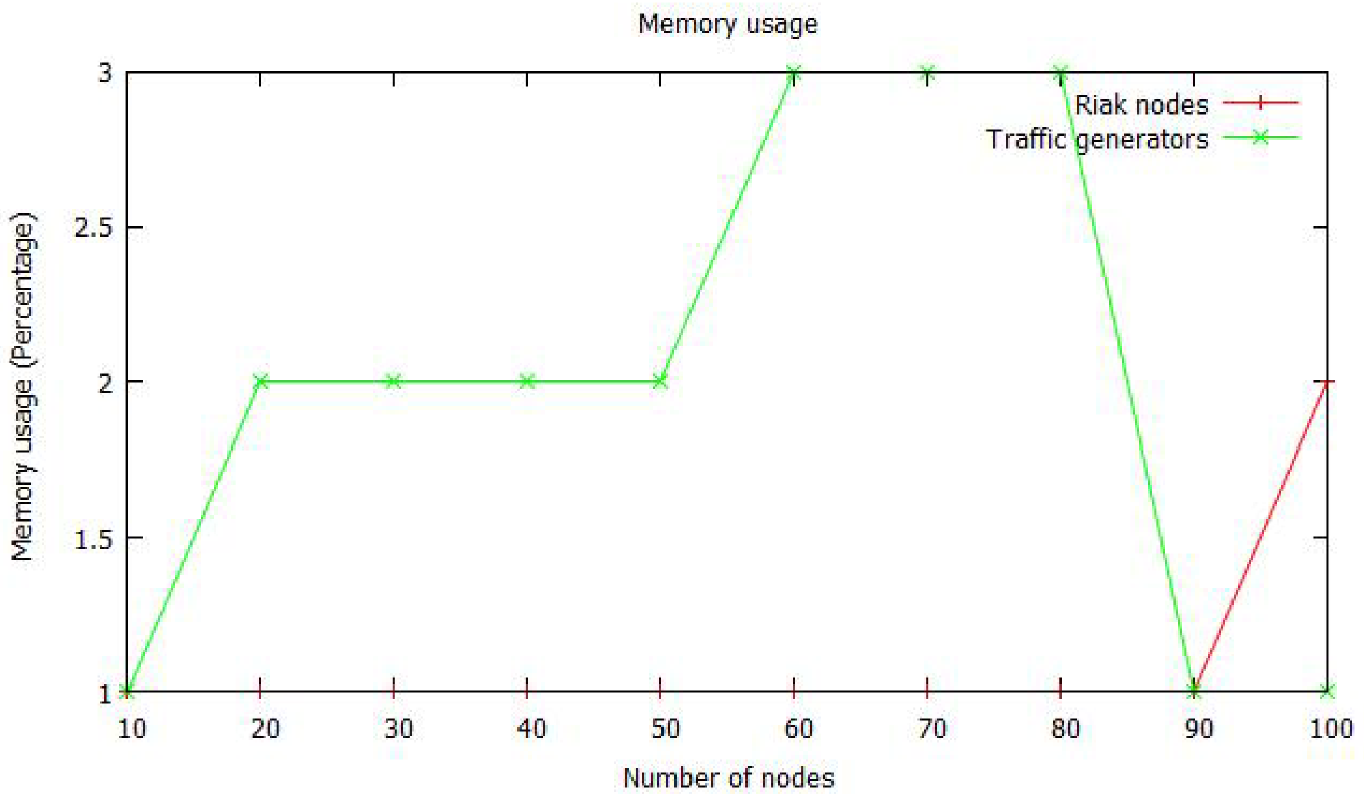
CPU Usage

Profiling Resource Usage



DISK Usage

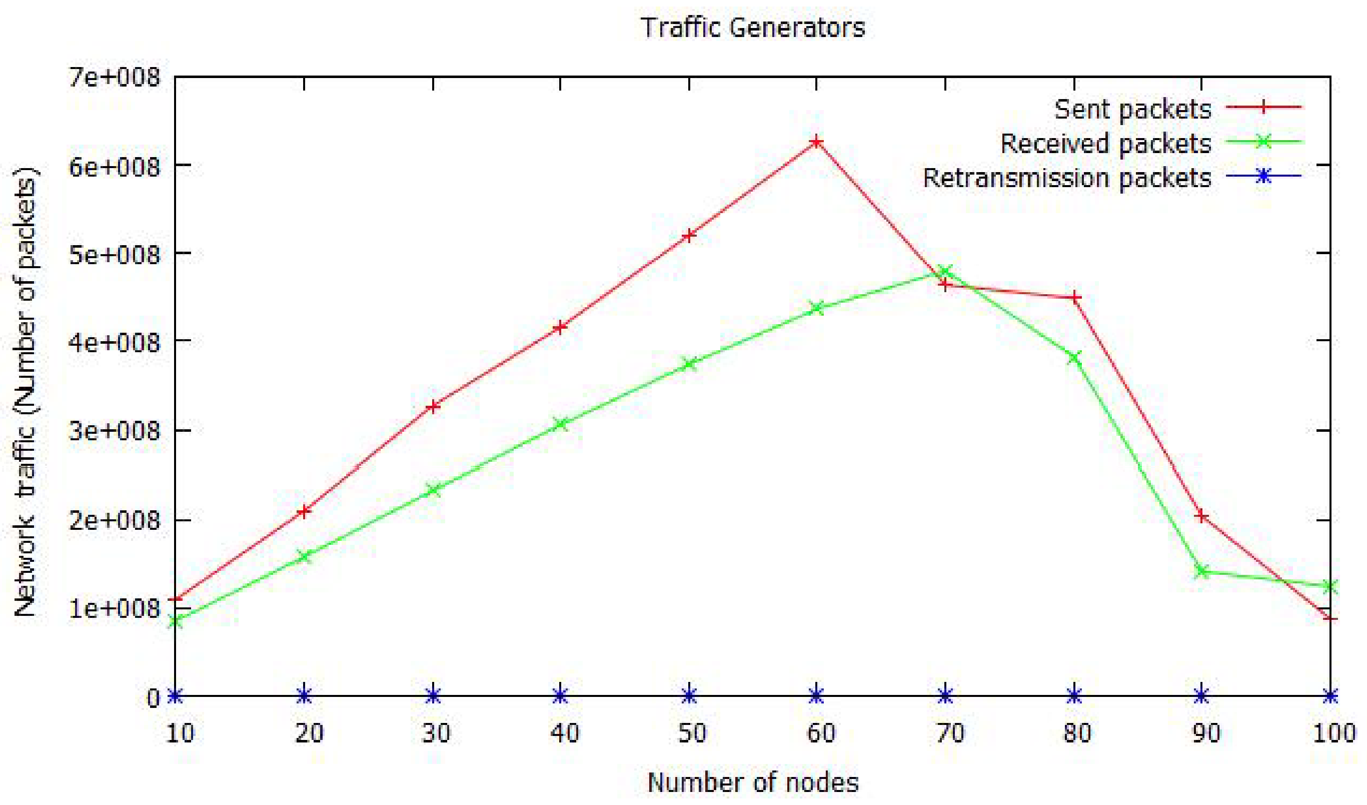
Profiling Resource Usage



Memory Usage

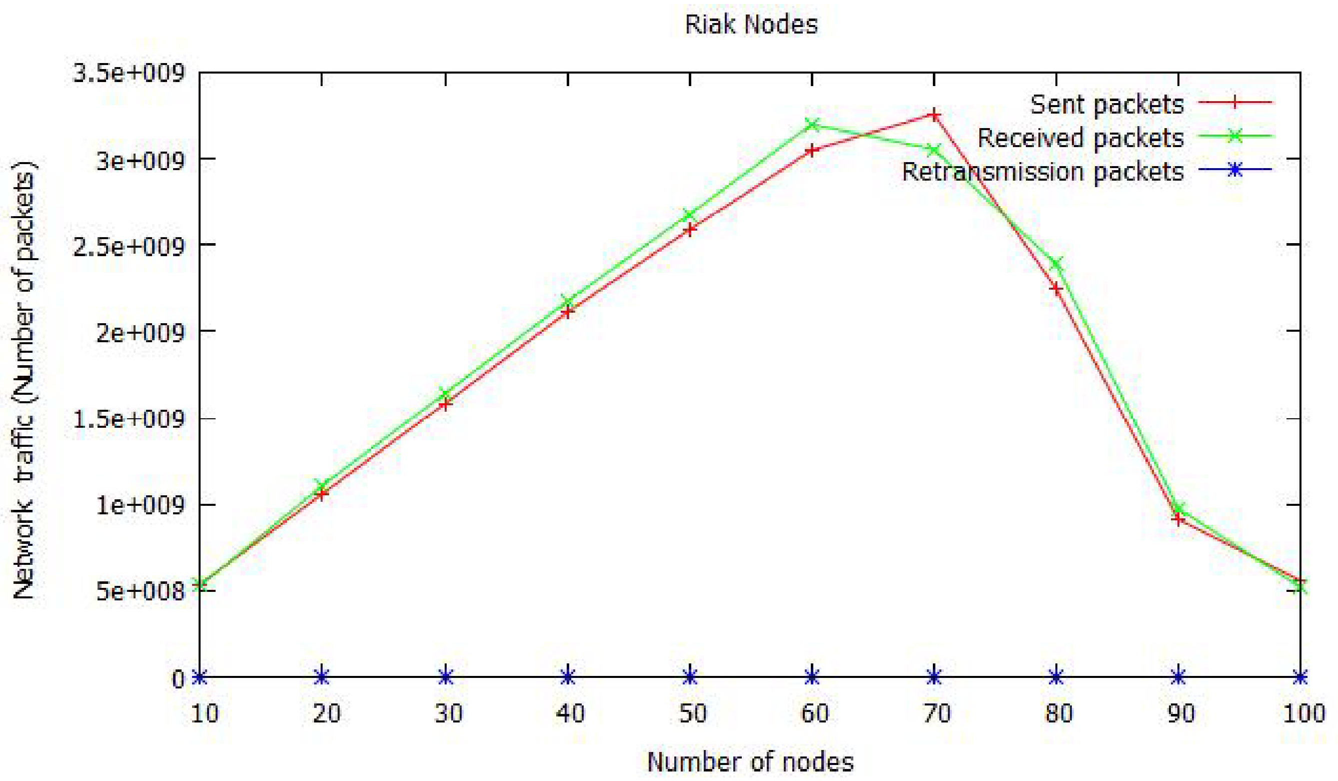
RELEASE

Profiling Resource Usage



Network Traffic of Generator Nodes

Profiling Resource Usage



Network Traffic of Riak Nodes

RELEASE

Bottleneck for Riak Scalability

CPU, *RAM*, *Disk*, and *Network* profiling reveal that they can't be bottleneck for Riak scalability.

Is the Riak scalability limits due to limits in distributed Erlang?

To find out, let’s measure the scalability of distributed Erlang.

RELEASE

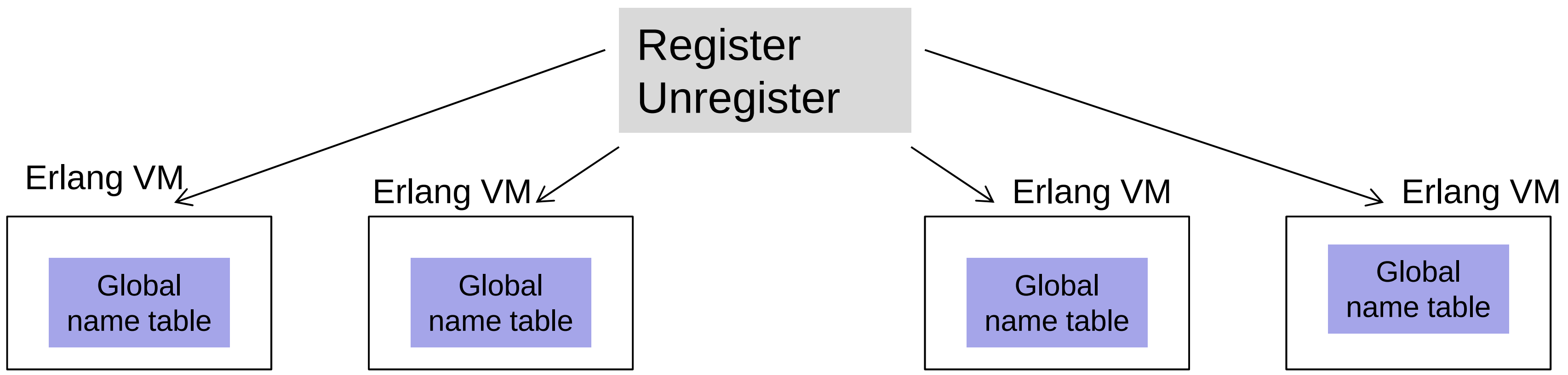
DEbench

- DEbench: a benchmarking tool for distributed Erlang
- It is based on Basho Bench
- Measures the throughput of a cluster of Erlang nodes
- Records the latency of distributed Erlang commands individually

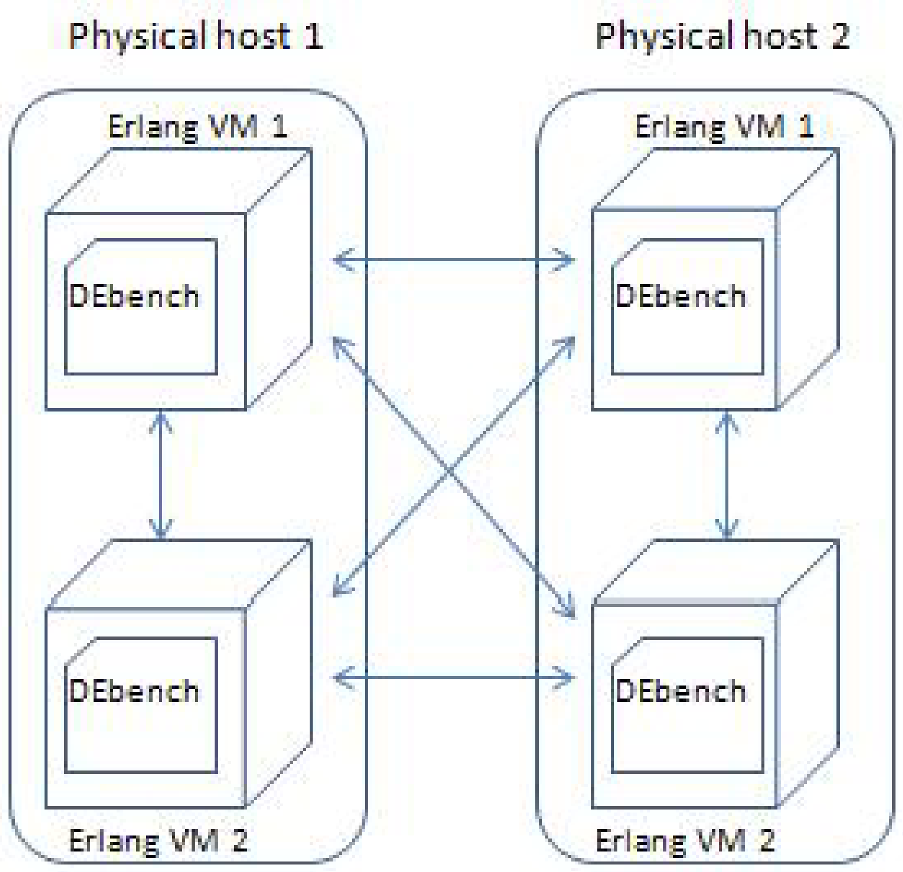
RELEASE

Distributed Erlang Commands

- ***Spawn/RPC:*** peer to peer commands
- ***register_name*** : global name tables located on every node
- ***unregister_name*** : global name tables located on every node
- ***whereis_name*** : a lookup in the local table

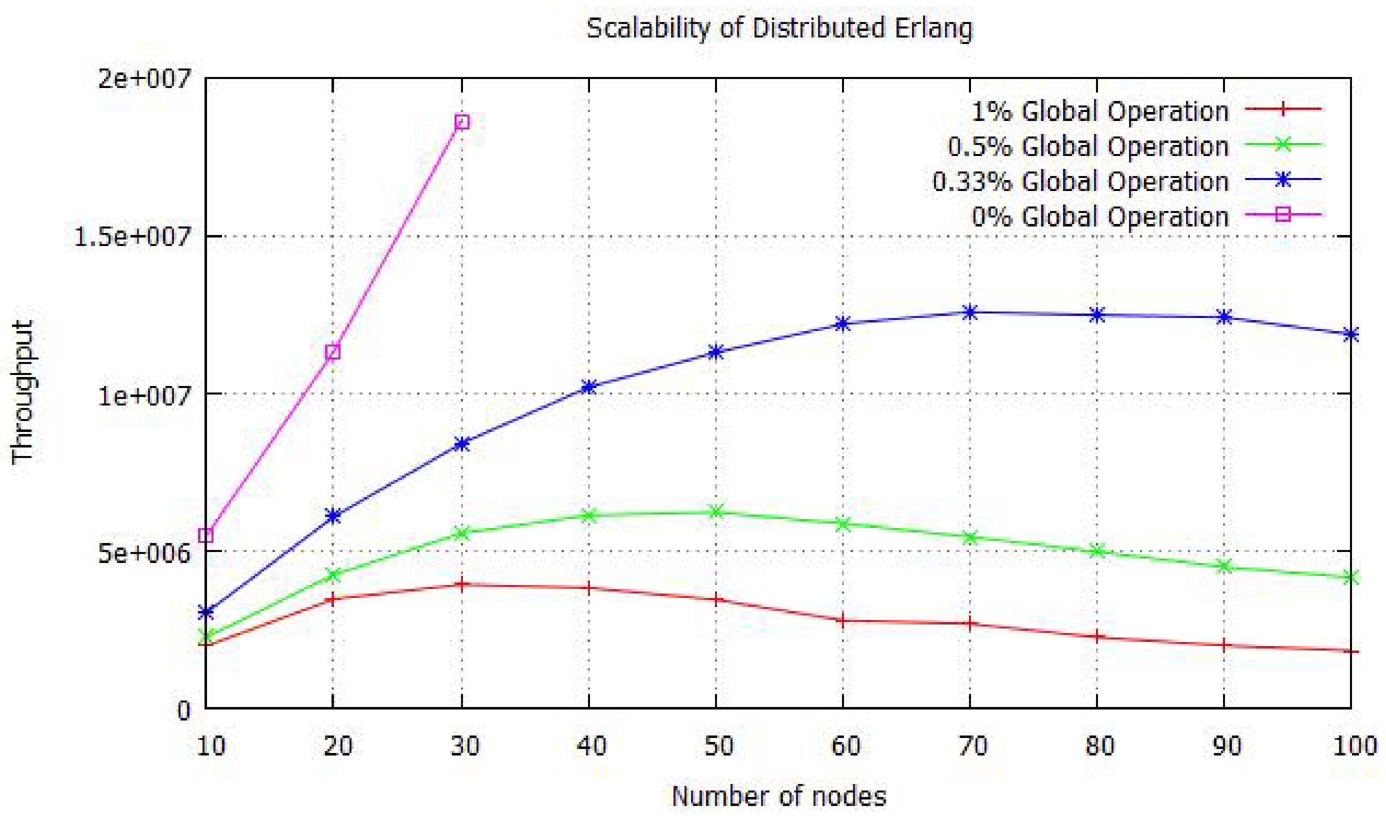


DEbench P2P Nodes



RELEASE

Frequency of Global Operation



Frequently	Max Throughput
1%	30 nodes
0.5%	50 nodes
0.33%	70 nodes
0%	1600 nodes

Global Operations limit the scalability of distributed Erlang

RELEASE

Riak Software Scalability

- Monitoring *global.erl* module from OTP library shows that Riak does NOT use any global operation.

- Instrumenting gen_server.erl module reveals that:
  - Of the 15 most time-consuming operations, only the time of *rpc:call* grows with cluster size.
  - Moreover, of the five Riak RPC calls, only *start_put_fsm* function from module *riak_kv_put_fsm_sup* grows with cluster size.

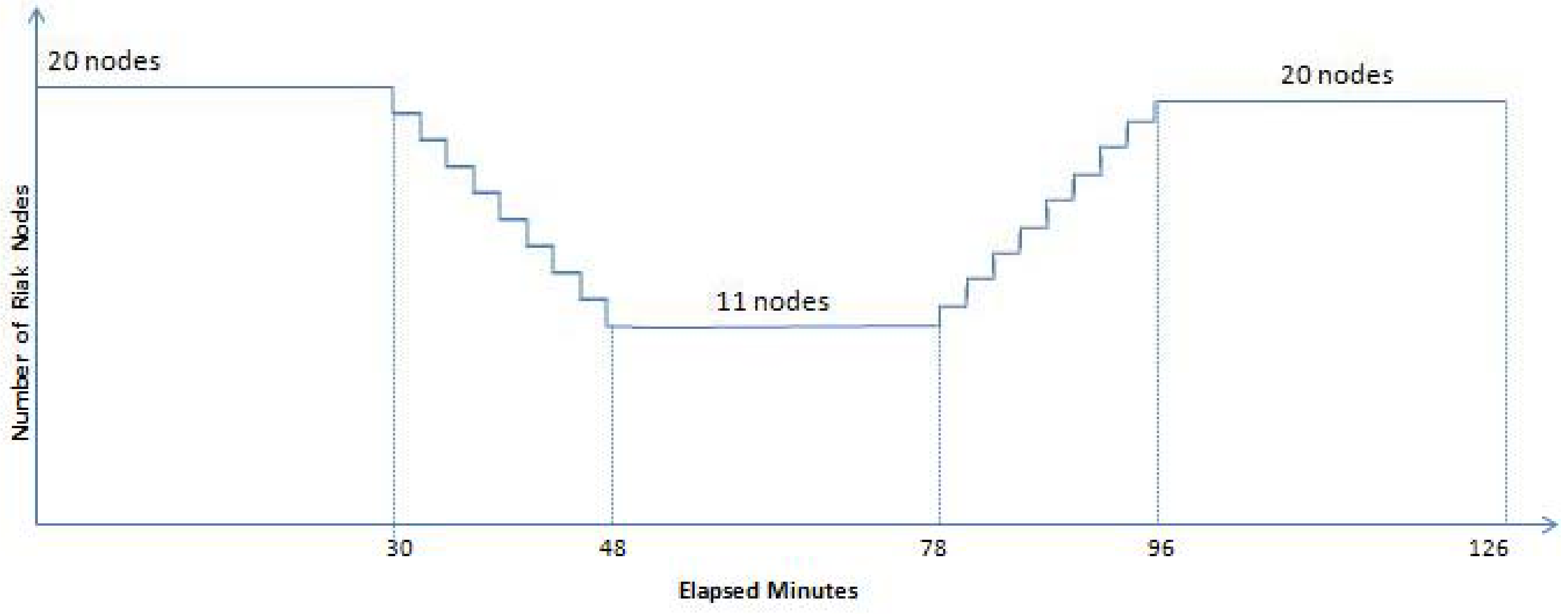
Eliminating the Bottlenecks

- Independently, Basho identified that two supervisor processes, i.e. *riak_kv_get/put_fsm_sup*, become bottleneck under heavy load, exhibiting build up in message queue length.

- To improve the Riak scalability in version 1.3 and 1.4 Basho applied a number of techniques and introduced new library sidejob (https://github.com/basho/sidejob).

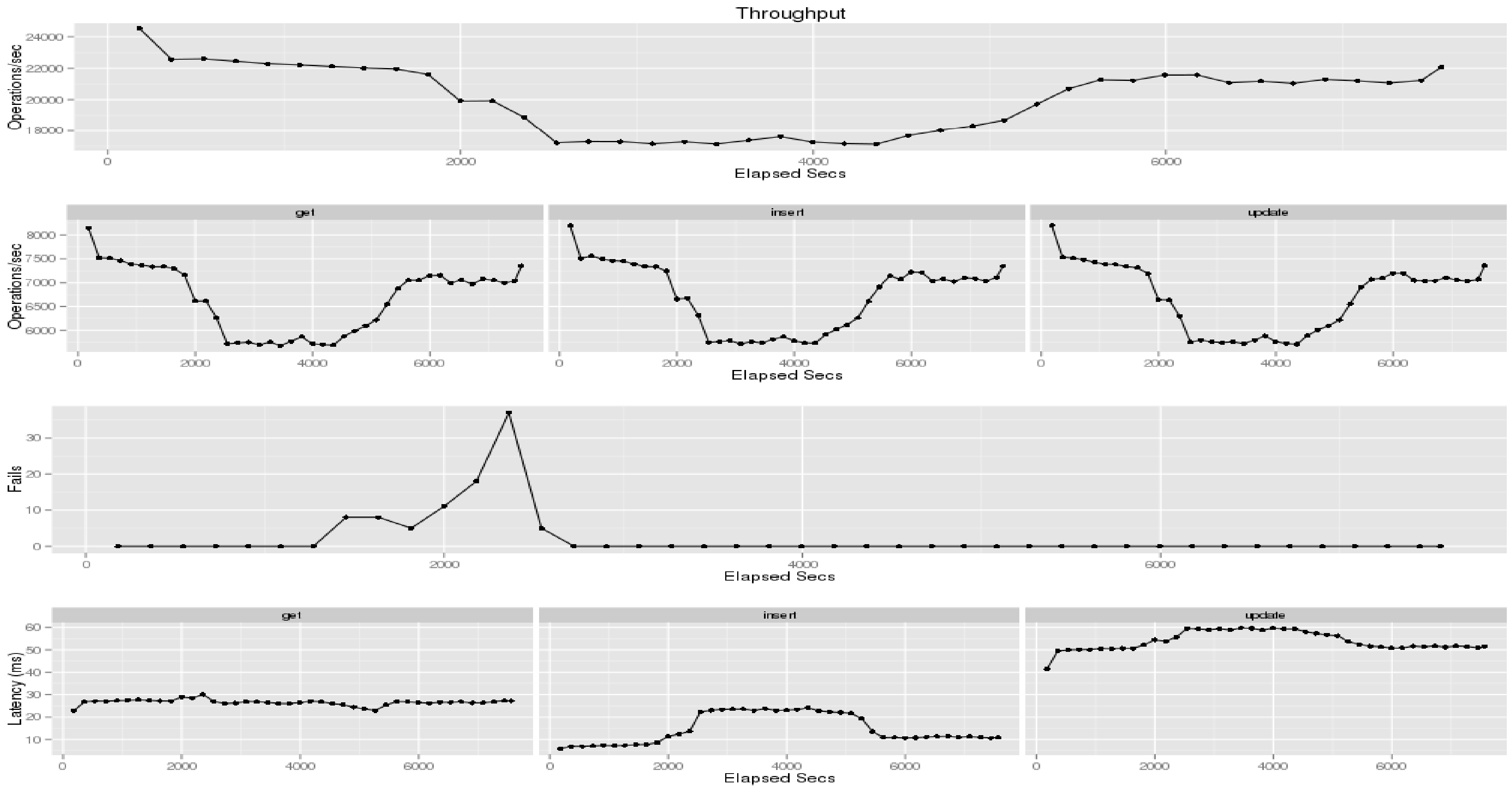
RELEASE

Riak1.1.1 Elasticity



Time-line shows Riak cluster losing and gaining nodes

Riak1.1.1 Elasticity



How Riak cluster deals with nodes leaving and joining

RELEASE

Observation

• Number of failures (37)
• Number of successful operations (approximately 3.41 million)

•When failed nodes come back up, the throughput has grown that shows Riak1.1.1 has a good **elasticity**.

Conclusion and Future work

✓ Our benchmark confirms that Riak has a good elasticity.

✓ We establish for the first time scientifically the scalability limit of Riak 1.1.1 as 60 nodes.

✓ We have shown how global operations limits the scalability of distributed Erlang.

✓Riak scalability bottelnecks are eliminated in Riak versions 1.3 and upcoming versions.

✓ In RELEASE, we are working to scale up distributed Erlang by grouping nodes in smaller partitions.

RELEASE

References

•Basho Bench
 http://docs.basho.com/riak/latest/ops/building/benchmarking/

•DE-Bench https://github.com/amirghaffari/DEbench

•A. Ghaffari, N. Chechina, P. Trinder, and J. Meredith. Scalable Persistent Storage for Erlang: Theory and Practice.

•N. Chechina, P. Trinder, A. Ghaffari, R. Green, K. Lundin, and R. Virding. The Design of Scalable Distributed Erlang.

•KalKyl Cluster http://www.uppmax.uu.se/the-kalkyl-cluster

•Sidejob https://github.com/basho/sidejob

RELEASE