SpringFS: Bridging Agility and Performance in Elastic Distributed Storage
Lianghong Xu, James Cipar, Elie Krevat, Alexey Tumanov, Nitin Gupta, Gregory Ganger (Carnegie Mellon University), Michael Kozuch (Intel Labs Pittsburgh)

MOTIVATION

- Cloud storage can and should be elastic
- Ability to extract/re-integrate servers on demand
- Elasticity is most useful when it is “agile”
- Agility: quickness of elastic resizing
- Value: machine-hour (money) savings
- Challenge: Data migration is expensive

Cloud storage can and should be elastic
- Ability to extract/re-integrate servers on demand
- Elasticity is most useful when it is “agile”
- Agility: quickness of elastic resizing
- Value: machine-hour (money) savings
- Challenge: Data migration is expensive

SPRINGFS DATA LAYOUT

- Continuum between “Rabbit” and “Sierra”
- Elasticity of Rabbit
- Peak write performance of Sierra
- Maximized agility along continuum between best cases

Results with industrial traces

- SpringFS achieves “close-to-ideal” machine hour usage
- Better than Rabbit when extracting servers
- Better than Sierra when re-integrating servers

RESULTS WITH INDUSTRIAL TRACES

- SpringFS achieves “close-to-ideal” machine hour usage
- Better than Rabbit when extracting servers
- Better than Sierra when re-integrating servers

Facebook HDFS trace

- Agile resizing → 50% less machine hour usage
- State-of-the-art elastic storage designs
- Sierra and Rabbit force painful tradeoff between elasticity, performance and agility
- Need a new elastic storage design that
- Fills the gap in the tradeoff space
- Achieves great agility
- Maintains performance and elasticity goals

BRIDGING AGILITY & PERFORMANCE

- Agile resizing → 50% less machine hour usage
- State-of-the-art elastic storage designs
- Sierra and Rabbit force painful tradeoff between elasticity, performance and agility
- Need a new elastic storage design that
- Fills the gap in the tradeoff space
- Achieves great agility
- Maintains performance and elasticity goals

RESULTS WITH INDUSTRIAL TRACES

- SpringFS achieves “close-to-ideal” machine hour usage
- Better than Rabbit when extracting servers
- Better than Sierra when re-integrating servers

Machine hour usage: 6-120% improvement

Data migration: 9-208X improvement

Average throughput per server

Write throughput scales with offload set

Minimize cleanup work with varied offload set

Number of servers with at least some primary replicas

Number of active servers

Minutes

Facebook HDFS trace

30 nodes, each with a 2GB file, 128MB block size

N: Total size of the cluster
p: Number of primaries of Rabbit
m: “Offload set” in SpringFS
- Bounds the offloading of primary replicas
- Adjustable tradeoff between write perf & cleanup work
- Adapted to workload changes

CONTINUUM BETWEEN “RABBIT” AND “SIERRA”

- Elasticity of Rabbit
- Peak write performance of Sierra
- Maximized agility along continuum between best cases

Maximum write performance (normalized to a single server)

criptive offloading interval
- Inherently offloading interval

SPRINGFS PERFORMANCE & CLEANUP WORK

- N: Total size of the cluster
- p: Number of primaries of Rabbit
  \[ p = \frac{N}{e^2} \] (e is Euler Constant)
- m: “Offload set” in SpringFS
  - Bounds the offloading of primary replicas
  - Adjustable tradeoff between write perf & cleanup work
  - Adapted to workload changes

- Cloud storage can and should be elastic
- Ability to extract/re-integrate servers on demand
- Elasticity is most useful when it is “agile”
- Agility: quickness of elastic resizing
- Value: machine-hour (money) savings
- Challenge: Data migration is expensive

- Agile resizing → 50% less machine hour usage
- State-of-the-art elastic storage designs
- Sierra and Rabbit force painful tradeoff between elasticity, performance and agility
- Need a new elastic storage design that
- Fills the gap in the tradeoff space
- Achieves great agility
- Maintains performance and elasticity goals

- Agile resizing → 50% less machine hour usage
- State-of-the-art elastic storage designs
- Sierra and Rabbit force painful tradeoff between elasticity, performance and agility
- Need a new elastic storage design that
- Fills the gap in the tradeoff space
- Achieves great agility
- Maintains performance and elasticity goals

Machine hour usage: 6-120% improvement

Data migration: 9-208X improvement

Average throughput per server

Write throughput scales with offload set

Minimize cleanup work with varied offload set

Number of servers with at least some primary replicas

Number of active servers

Minutes

Facebook HDFS trace

30 nodes, each with a 2GB file, 128MB block size

N: Total size of the cluster
p: Number of primaries of Rabbit
m: “Offload set” in SpringFS
- Bounds the offloading of primary replicas
- Adjustable tradeoff between write perf & cleanup work
- Adapted to workload changes

CONTINUUM BETWEEN “RABBIT” AND “SIERRA”

- Elasticity of Rabbit
- Peak write performance of Sierra
- Maximized agility along continuum between best cases

Maximum write performance (normalized to a single server)

criptive offloading interval
- Inherently offloading interval

SPRINGFS PERFORMANCE & CLEANUP WORK

- N: Total size of the cluster
- p: Number of primaries of Rabbit
  \[ p = \frac{N}{e^2} \] (e is Euler Constant)
- m: “Offload set” in SpringFS
  - Bounds the offloading of primary replicas
  - Adjustable tradeoff between write perf & cleanup work
  - Adapted to workload changes

- Cloud storage can and should be elastic
- Ability to extract/re-integrate servers on demand
- Elasticity is most useful when it is “agile”
- Agility: quickness of elastic resizing
- Value: machine-hour (money) savings
- Challenge: Data migration is expensive

- Agile resizing → 50% less machine hour usage
- State-of-the-art elastic storage designs
- Sierra and Rabbit force painful tradeoff between elasticity, performance and agility
- Need a new elastic storage design that
- Fills the gap in the tradeoff space
- Achieves great agility
- Maintains performance and elasticity goals

Machine hour usage: 6-120% improvement

Data migration: 9-208X improvement

Average throughput per server

Write throughput scales with offload set

Minimize cleanup work with varied offload set

Number of servers with at least some primary replicas

Number of active servers

Minutes

Facebook HDFS trace

30 nodes, each with a 2GB file, 128MB block size

N: Total size of the cluster
p: Number of primaries of Rabbit
m: “Offload set” in SpringFS
- Bounds the offloading of primary replicas
- Adjustable tradeoff between write perf & cleanup work
- Adapted to workload changes

CONTINUUM BETWEEN “RABBIT” AND “SIERRA”

- Elasticity of Rabbit
- Peak write performance of Sierra
- Maximized agility along continuum between best cases

Maximum write performance (normalized to a single server)

criptive offloading interval
- Inherently offloading interval