To Waffinity and Beyond:
A Scalable Architecture for Incremental Parallelization of File System Code

Matthew Curtis-Maury, PhD
Vinay Devadas, PhD
Vania Fang
Aditya Kulkarni

NetApp, Inc

© 2016 NetApp, Inc. All rights reserved.

NetApp

Background

- Data ONTAP is a storage operating system
- WAFL File System processes operations in the form of messages
- Competitive performance requires CPU scaling
  - WAFL is millions of lines of complicated code
  - A pure locking model is impractical
  - Many other techniques in the literature
    - Barrelfish, fos, Corey, Multikernel, …

© 2016 NetApp, Inc. All rights reserved.

WAFL Parallelization Overview

- In the beginning… WAFL processed all messages sequentially

- WAFL parallelism leverages data partitioning

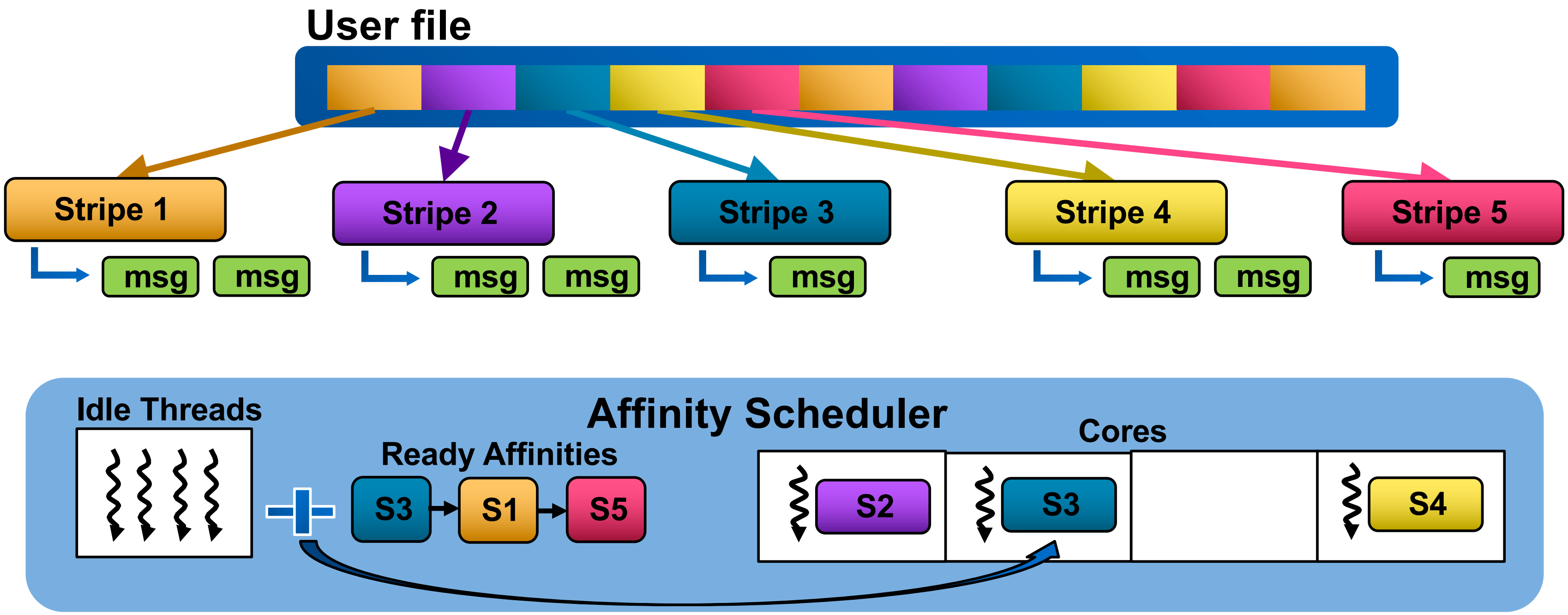
- Set of techniques to allow incremental parallelization
  - Classical Waffinity – Partition user files into chunks
  - Hierarchical Waffinity – Partition many FS data structures
  - Hybrid Waffinity – Add locking *within* the data partition framework

- These techniques have been implemented in our production OS and deployed on >200K systems

© 2016 NetApp, Inc. All rights reserved.

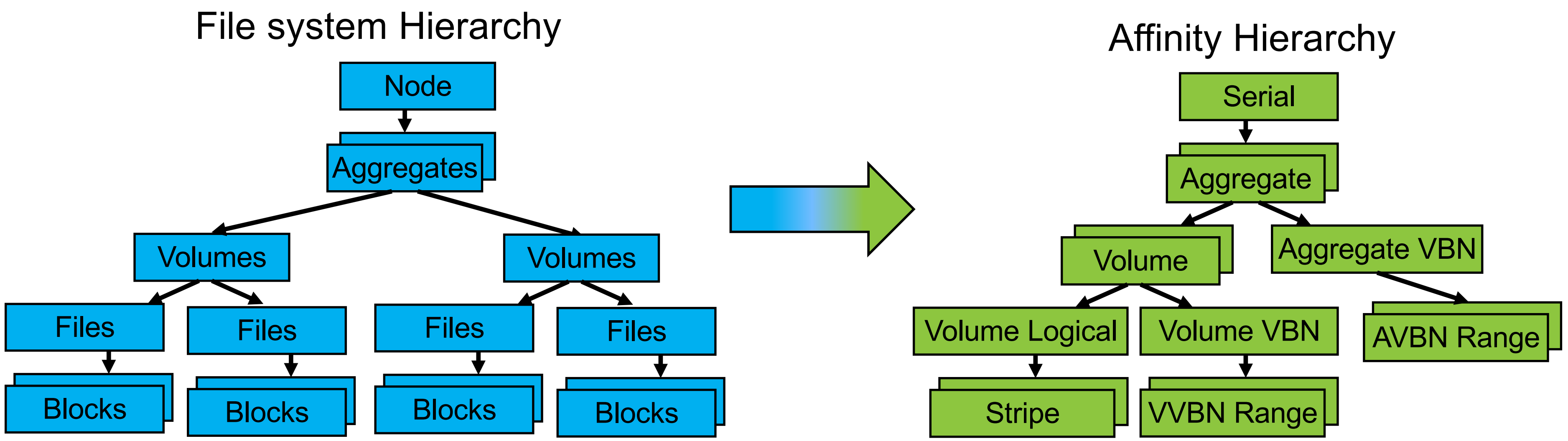
Classical Waffinity (2006)

- Partition user files into fixed-size chunks called *file stripes*
- Rotated over a set of message queues called *Stripe affinities*
- Affinity scheduler dynamically assigns affinities to threads
- Include a *Serial affinity* to process work outside of file stripes



© 2016 NetApp, Inc. All rights reserved.

Hierarchical Waffinity (2011)



- Hierarchical data partitioning to match hierarchical data
  - Particular shape fine-tuned for WAFL
  - Hierarchical permissions / exclusion
- Allows parallelization of work that used to run in Serial affinity
- Friendly to incremental parallelization

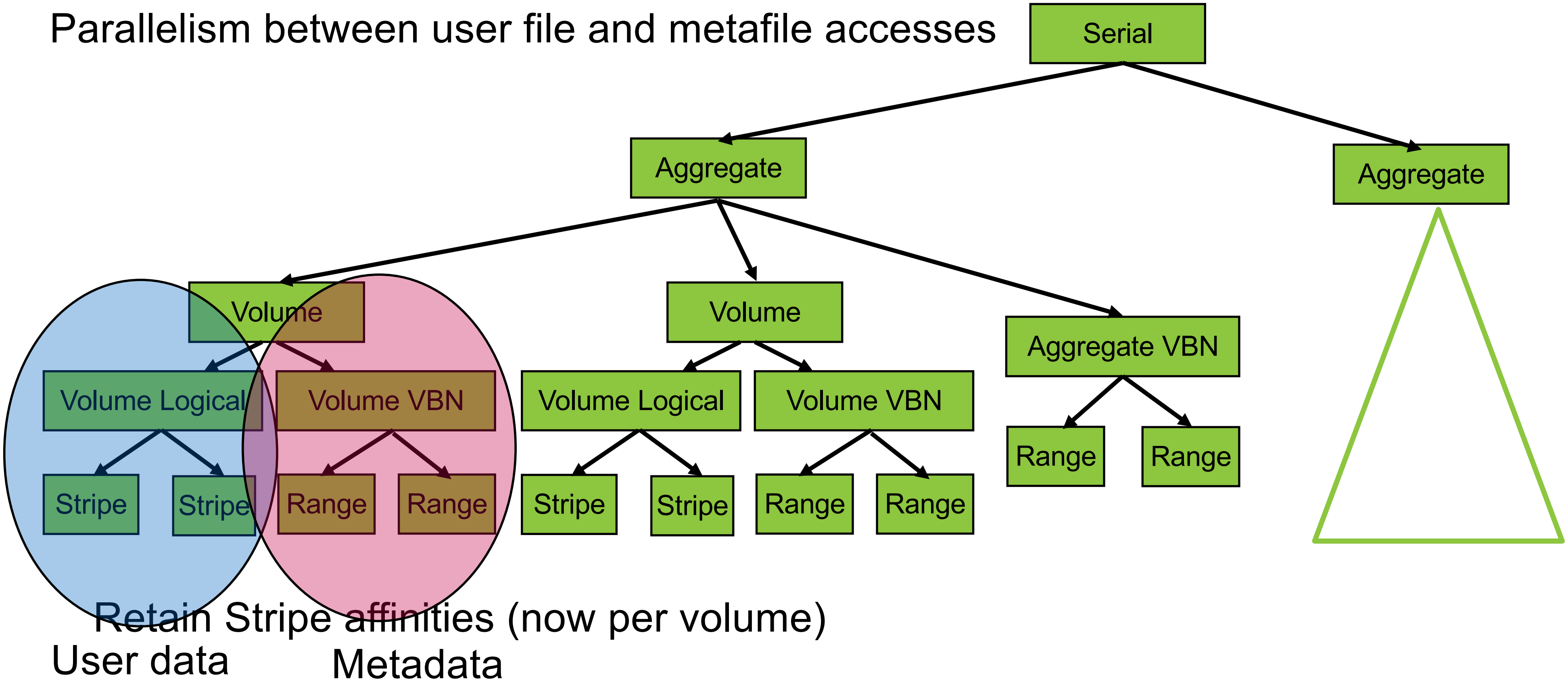
© 2016 NetApp, Inc. All rights reserved.

NetApp

Hierarchical Waffinity – Data mappings

Parallelism between different volumes and aggregates

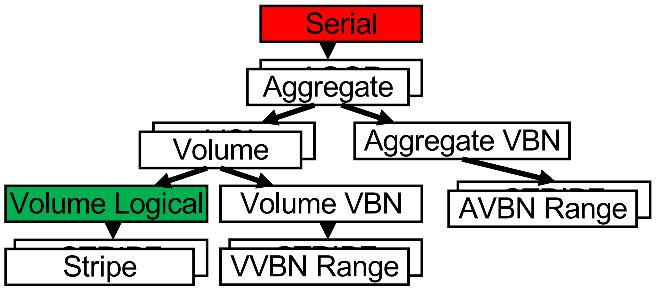
Parallelism between user file and metafile accesses

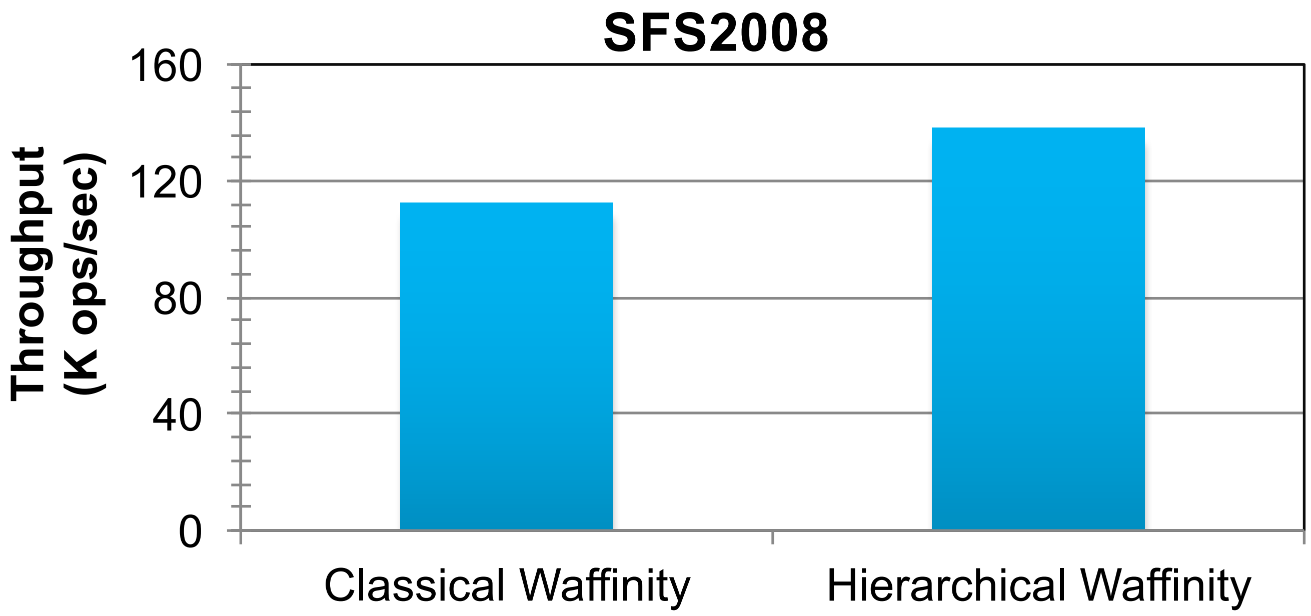


Retain Stripe affinities (now per volume)

© 2016 NetApp, Inc. All rights reserved.

Classical vs. Hierarchical Waffinity

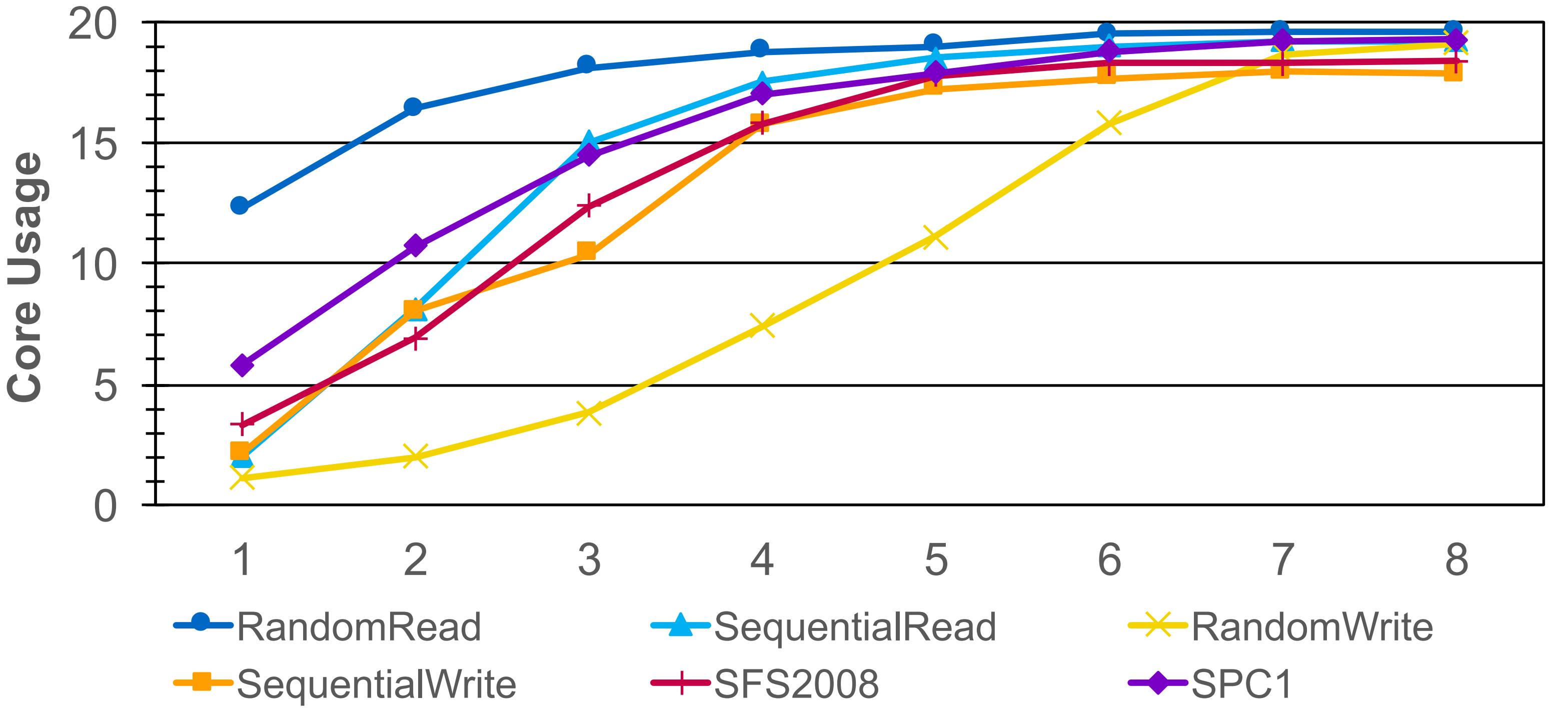




- SFS2008 contains metadata operations (Create, Remove, etc)
  - Classical Waffinity: Ran in Serial affinity (48% of wallclock time)
  - Hierarchical Waffinity allows the messages to run in Volume Logical
- **~3 additional cores** used translated into a **23% throughput increase**

© 2016 NetApp, Inc. All rights reserved.

Hierarchical Waffinity CPU Scaling



- 95% average core occupancy across 6 key workloads

© 2016 NetApp, Inc. All rights reserved.

NetApp

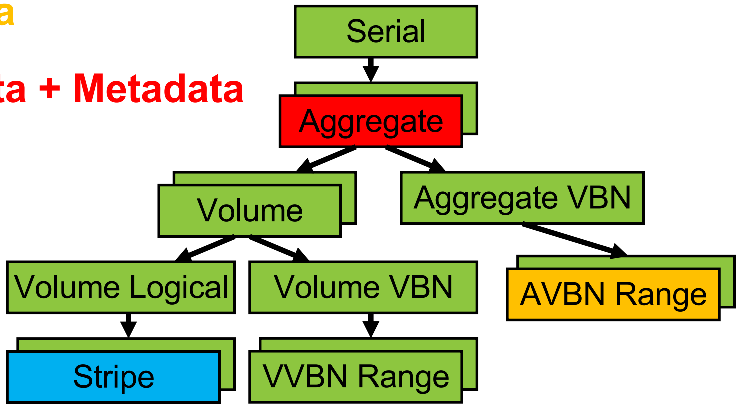
Hybrid Waffinity (2016)

- Some important workloads access two different file blocks
  - Mappings optimized for traditional cases not well-suited here
- Hybrid Waffinity combines partitioning with fine-grained locking
  - **Particular blocks** are protected with locking from multiple affinities
  - Continues to allow **incremental** development

User Data

Metadata

User Data + Metadata



Serial

Aggregate

Volume

Aggregate VBN

Volume Logical

Volume VBN

AVBN Range

Stripe

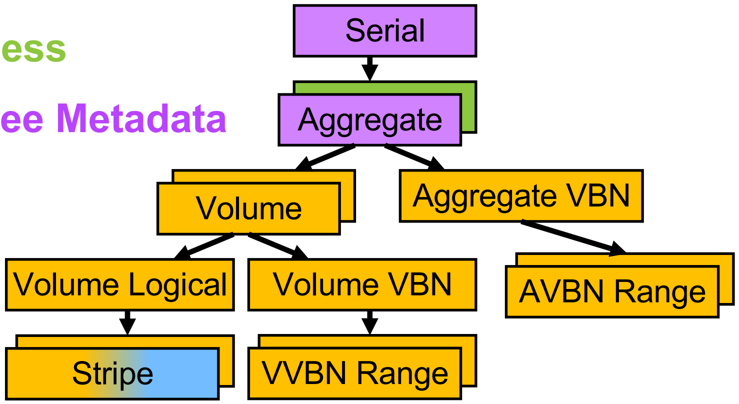
VVBN Range

Metadata with locking

User Data

No Access

Lock-free Metadata



Serial

Aggregate

Volume

Aggregate VBN

Volume Logical

Volume VBN

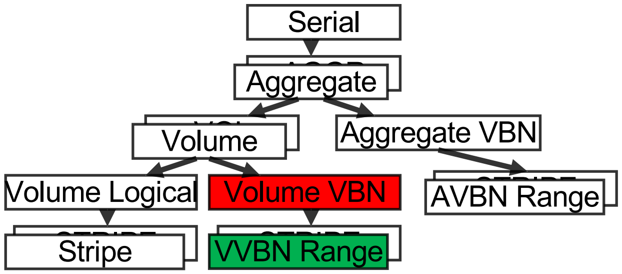
AVBN Range

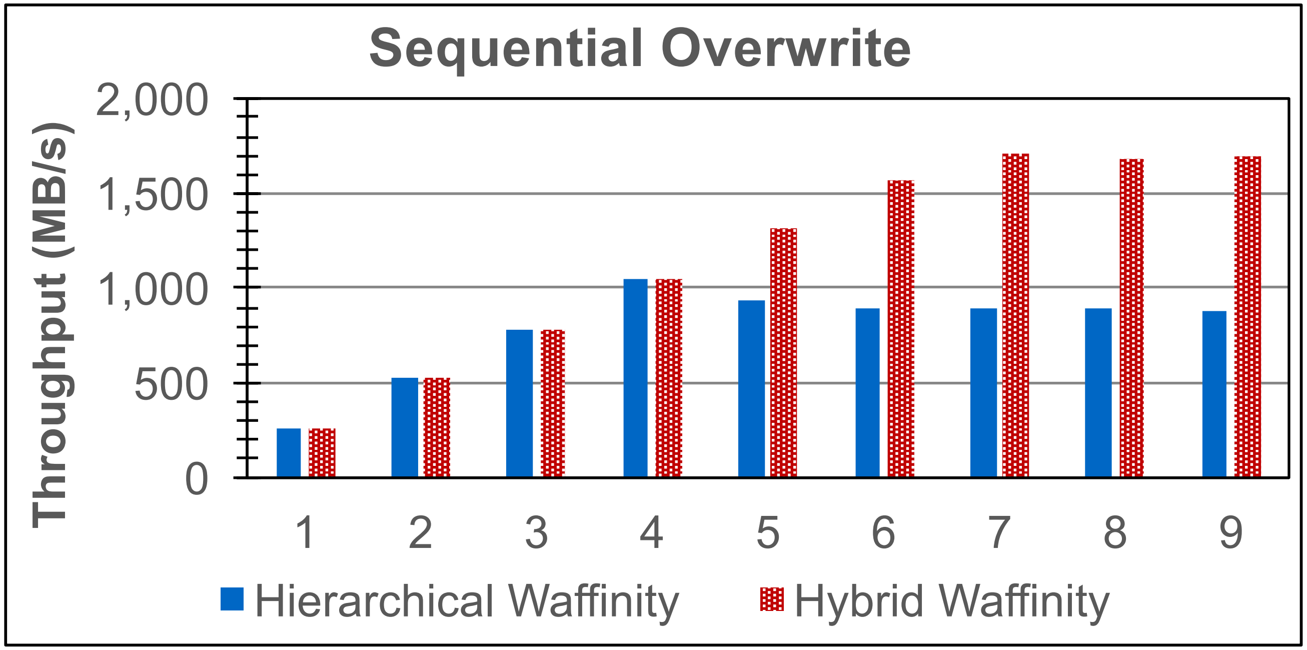
Stripe

VVBN Range

© 2016 NetApp, Inc. All rights reserved.

Hybrid vs. Hierarchical Waffinity





- Block free operations in Volume VBN for two metafile accesses
- Hybrid Waffinity parallelizes it further into VVBN Range
- **6 additional cores** translated into a **91% throughput increase**

© 2016 NetApp, Inc. All rights reserved.

Conclusion

- Developed a set of techniques to allow incremental parallelization of the WAFL file system
  - Focused on data partitioning
  - Selectively added in locking in a restricted way

- Provided insight into the internals of WAFL

© 2016 NetApp, Inc. All rights reserved.

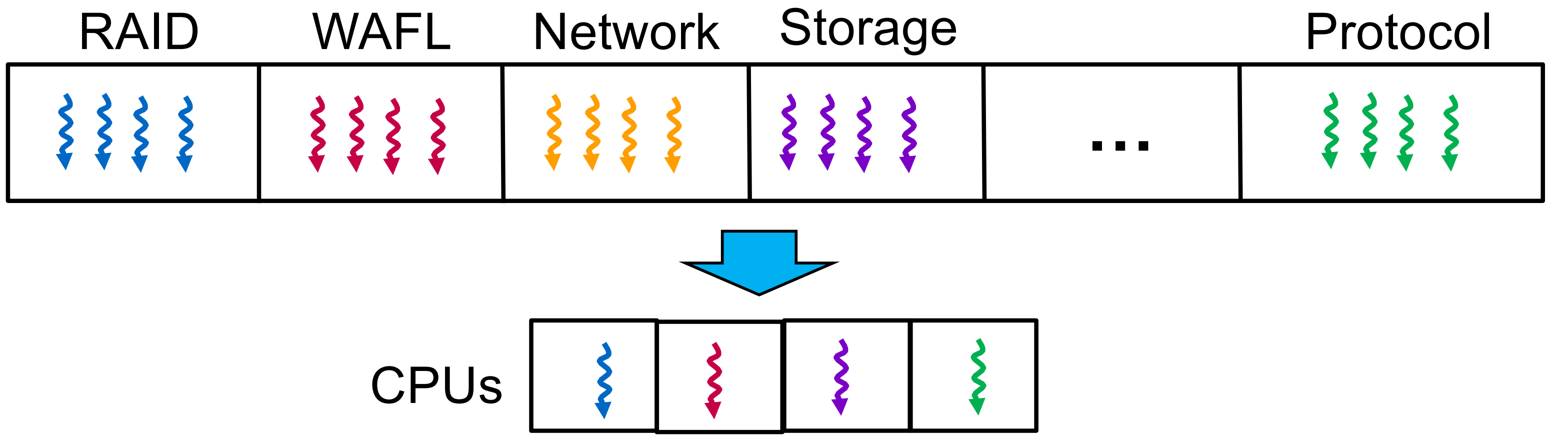
Thank you.

© 2016 NetApp, Inc. All rights reserved

History of Parallelism in ONTAP

- Data ONTAP was created for single-CPU systems of 1994

- Parallelism via “Coarse-grained Symmetric Multi-processing”
  - Each subsystem was assigned to a single-threaded *domain*
  - Minimal explicit locking required, message passing between domains
  - Scaled to 4 cores, but all of WAFL serialized

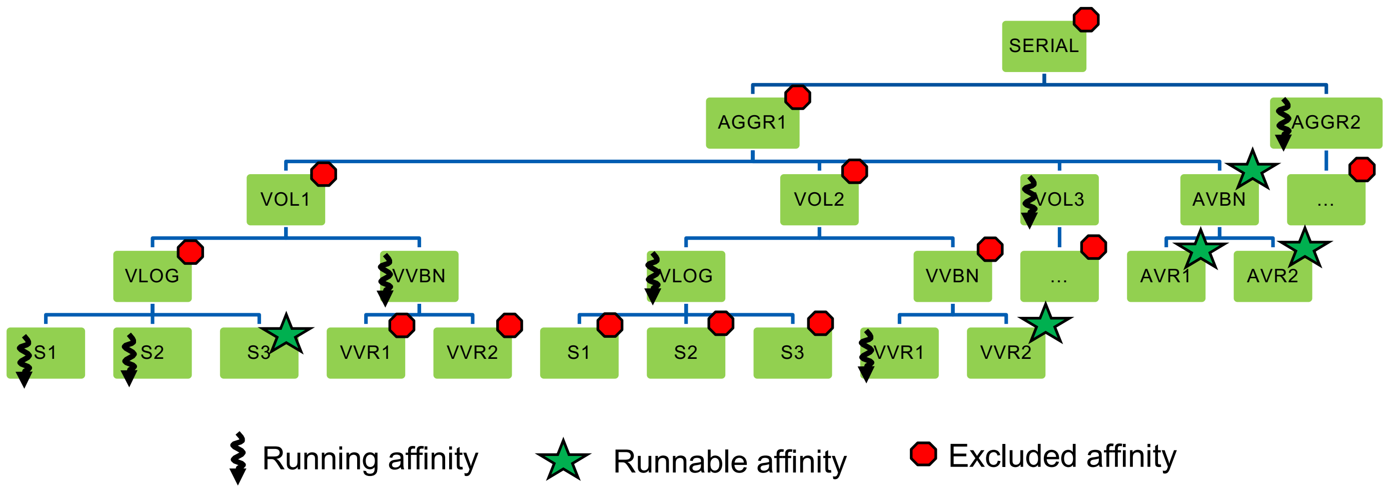


© 2016 NetApp, Inc. All rights reserved.

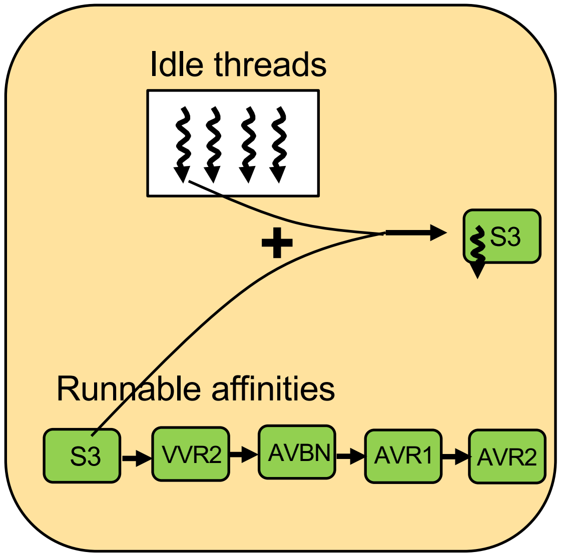
NetApp

Example Scheduler State

Affinity Hierarchy



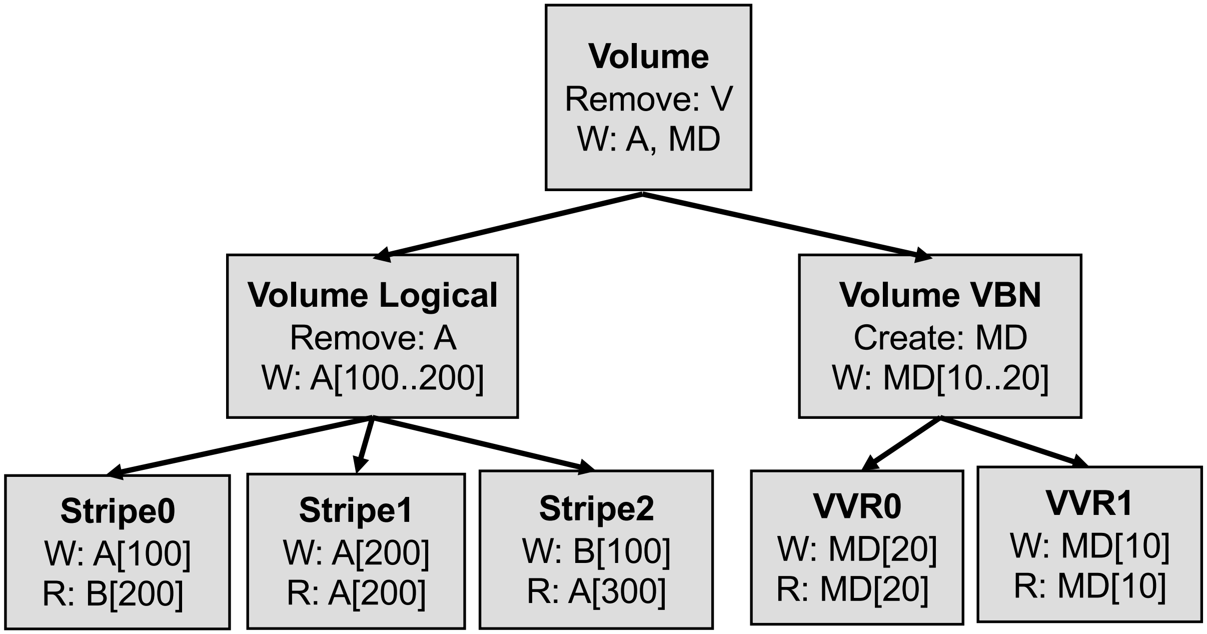
Hierarchical Scheduler



- Scheduler keeps FIFO list of *runnable* affinities
- Threads call into Affinity scheduler for work
- Work in *coarse* affinities starves the system of runnable affinities

© 2016 NetApp, Inc. All rights reserved.

Example Affinity Mappings



Volume
Remove: V
W: A, MD

Volume Logical
Remove: A
W: A[100..200]

Volume VBN
Create: MD
W: MD[10..20]

Stripe0
W: A[100]
R: B[200]

Stripe1
W: A[200]
R: A[200]

Stripe2
W: B[100]
R: A[300]

VVR0
W: MD[20]
R: MD[20]

VVR1
W: MD[10]
R: MD[10]

Development Experiences

- Hierarchical Waffinity
  - Parallelization occurs at the message granularity, changed O(hundreds) LoC
  - Only parallelize critical messages, in common paths, and to a suitable affinity
  - Infrastructure required 22k LoC

- Hybrid Waffinity
  - Infrastructure for each access mode was ~3k LoC
  - Using Eject and Insert is easy, fewer than 20 lines per message optimized
  - Write involves updating and restructuring message handler -> 2k LoC
  - Now applying to Inodes with modest code changes

© 2016 NetApp, Inc. All rights reserved.