THIC Inc. The Premier Advanced Storage Technology Forum

Tornado Codes for Archival Storage

Matthew Woitaszek

NCAR Computational Science, 1850 Table Mesa Dr, Boulder CO 80305-5602

E-mail: matthew@ucar.edu

Presented at the THIC Meeting at the National Center for Atmospheric Research, 1850 Table Mesa Drive, Boulder CO 80305-5602

July 19, 2006

Tornado Codes for Archival Storage

Matthew Woitaszek
NCAR Computational Science
1850 Table Mesa Drive
Boulder, CO 80305

Phone: 303/497-1279
Fax: 303/497-1286
E-mail: matthew@ucar.edu

Presented at the THIC meeting at
NCAR, Boulder CO, 19 July 2006

NCAR

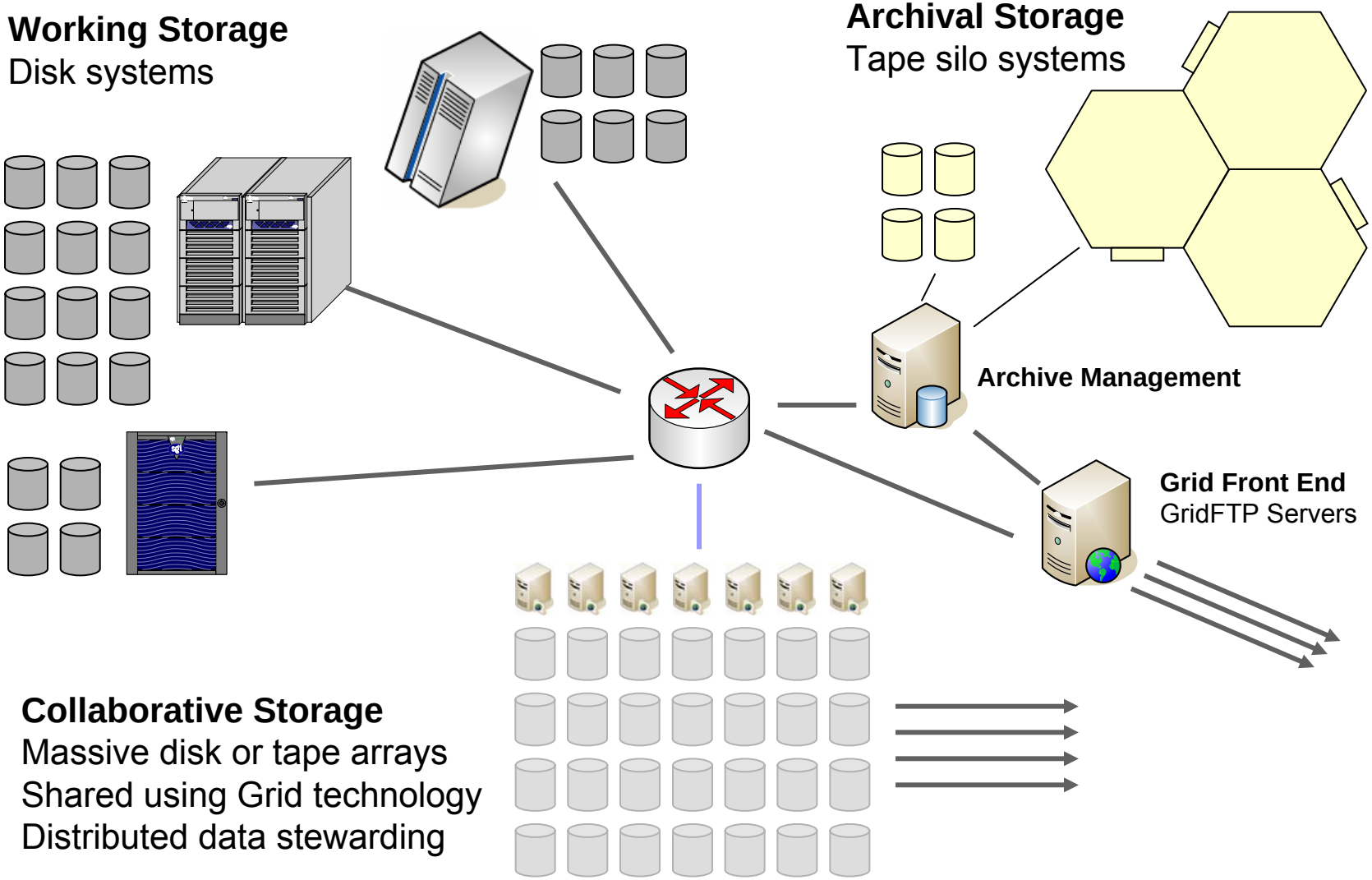
Key Points

- Tornado Codes can provide fault tolerant storage
  - Single site applications: better than RAID
  - Distributed applications: better than replication
  - Verify then trust
- Tornado codes can work with existing Data Grids
  - Low computational overhead
  - Behind-the-scenes server-side fault tolerance
  - No reason to alter interfaces, use existing Data Grid tools
- Tornado codes are an excellent match for MAID
  - Leverage redundant data
  - Optimize device mount count and limit power consumption

Outline

- Background and Motivation
- Experimental Method
- Fault Tolerance Results
  - Single Tornado Code Fault Tolerance
  - Distributed and Federated System Fault Tolerance
- Applications to MAID Systems
- Conclusions and Future Work

Storage for High Performance Computing

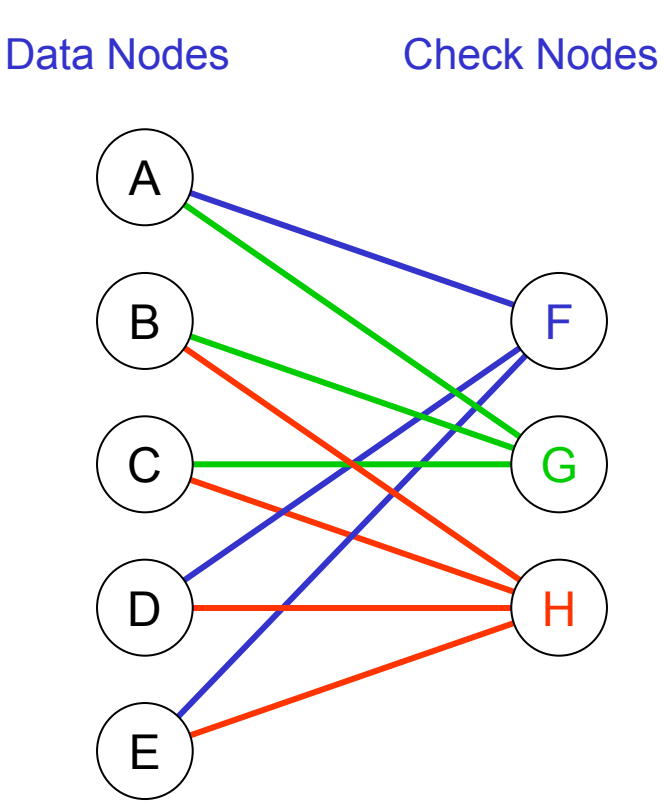


Motivation – Filesystem Features

- Typical desired features
  - Performance (no waiting)
  - Availability (no downtime)
  - Reliability (no data loss)
  - Inexpensive (no cost)
  - Scalability (no limits)

- Apply Tornado Codes to distributed archival storage
  - Prime directive: Never lose data
  - Optimize performance and resource utilization
  - Leverage existing legacy technology
  - Support emerging technologies and solutions (Grid, MAID)

LDPC Codes and Tornado Codes



Low Density Parity Check codes

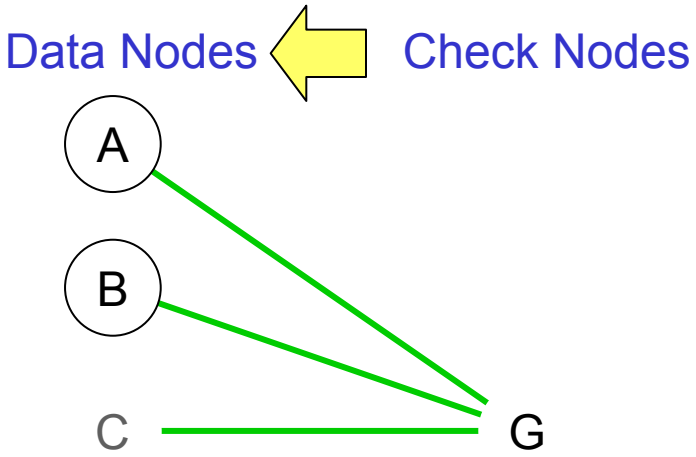
- Gallager, 1963
- Luby, 1990s

- Cascaded irregular LDPC graphs are Tornado Codes
  - Average degree of connectivity
  - Distribution of degree
  - Randomly generated

- Simple XOR operation has low computational overhead

Storage Applications – Safety and Speed

- Tornado Code advantages
  - Fault tolerance
  - Performance optimizations



Retrieve C or G?

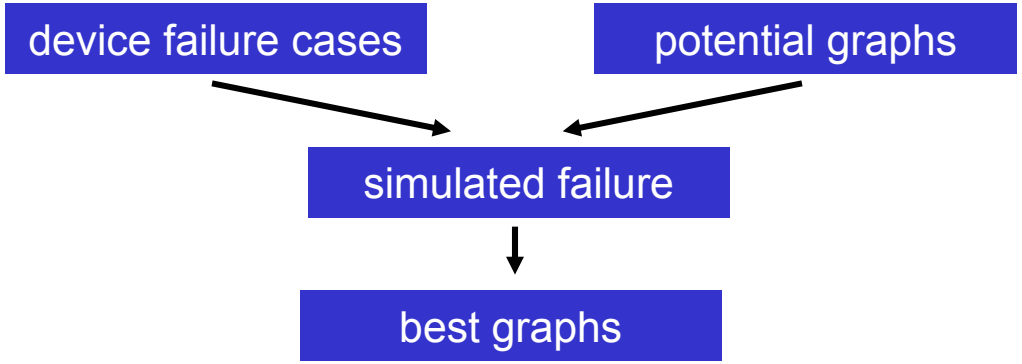
- If one node is dead, the decision is easy!
- If both nodes are available, choose the node with less waiting

Related Work

- Typhoon (Weatherspoon, 1999) – **Performance**
  - **Higher performance alternative to Reed-Solomon coding** for OceanStore's archival storage class
  - Simple algorithms for Tornado Code generation
- RobuSTore (Xia, 2006) – **Hide latency**
  - LDPC codes and **speculative access**
  - Hide latency from slow devices
- This work examines the risk of **data loss** in distributed filesystems using Tornado Codes

Experimental Method

- Construct and evaluate a Tornado Coding scheme



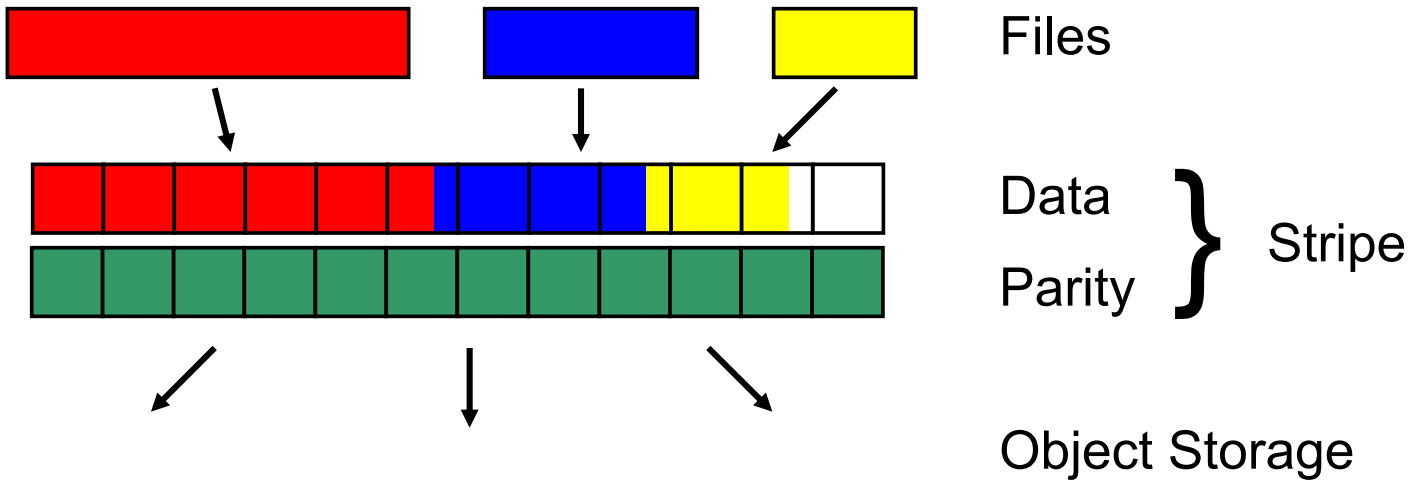
- Initial worst case failure identification
- Extensive final graph profiling
  - Bound “probabilistically successful” data reconstruction
  - Combine with empirical device failure rates for reliability
- Final graphs used for filesystem implementation

Evaluation Metrics

- Reconstruction Efficiency
  - **Overhead**: average data retrieval required to reconstruct
  - Example: 48 data and 48 check nodes
    - Minimum: 48 data nodes
    - Random: average of 62 nodes (1.29)
  - Other authors: between 1.2 and 1.4
- Fault Tolerance
  - **First failure**: How many nodes/devices can be lost?
  - **Reliability**: probability of failure over time
- Missing blocks may be…
  - available but not retrieved – reconstruction efficiency
  - permanently unavailable – fault tolerance

Testing Tornado Codes

- Fault tolerance analysis of 96-node graphs
  - Small enough to explicitly manage device state
  - Large enough to take advantage of selective retrievals
- Storing files using Tornado Code graphs
  - Per-file encoding
  - Using a log-based filesystem



Tornado Codes for Archival Storage

- **Single site fault tolerance**
- Federated system fault tolerance
- Optimization for MAID systems

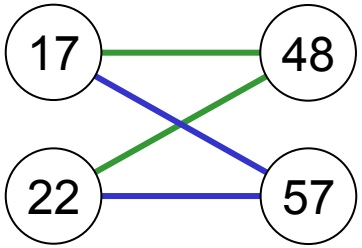
First Experiences with Tornado Codes

- Tornado code graphs fail to reconstruct due to **closed set of right nodes** (subgraph identification)

Failure with 2 lost nodes

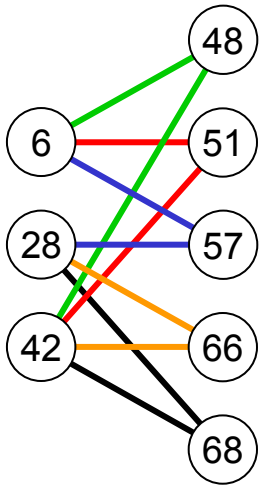
Left node [right nodes]

17 [48,57]
22 [48,57]



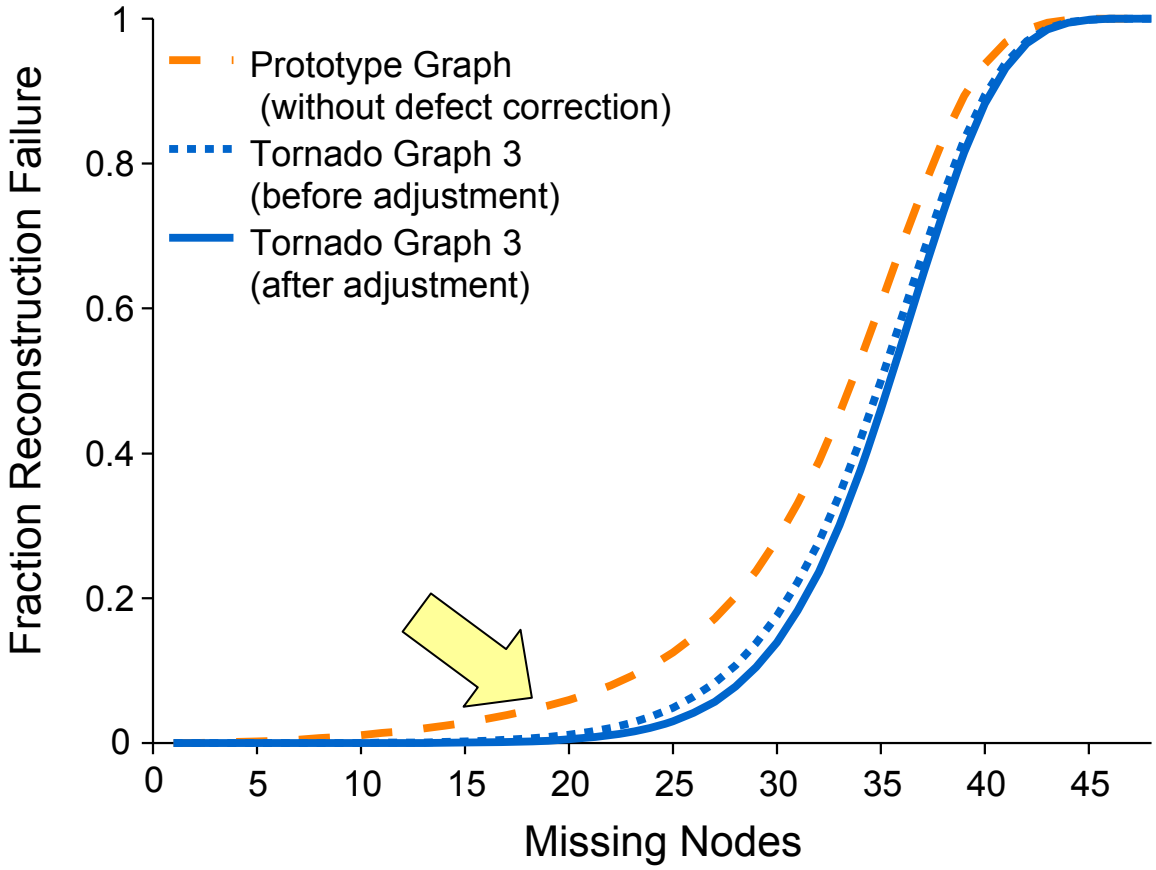
Failure with 3 lost nodes

6 [48,51,57]
28 [57,66,68]
42 [48,51,66,68]



- Possible to correct by expanding sets
- Example 96-node graph survives all 4-node failures and only 6 / 61,124,064 5-node cases fail

Tornado Code Defect Detection Results

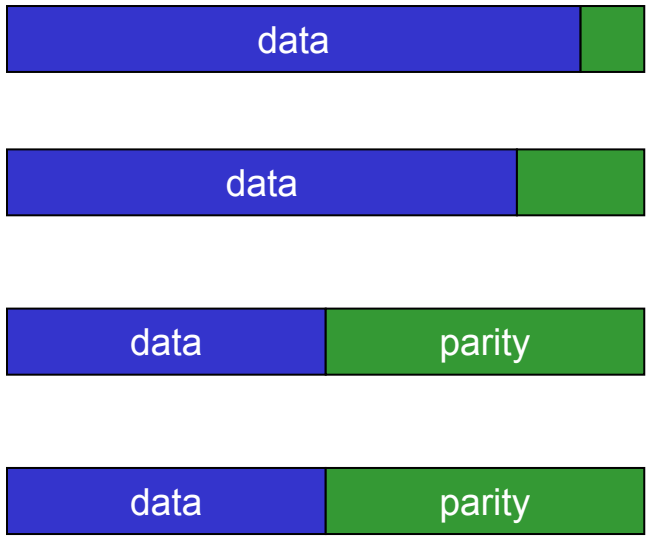


	First Failure
Prototype	2
Before	4
After	5

Never trust a randomly generated Tornado Code graph without testing first!

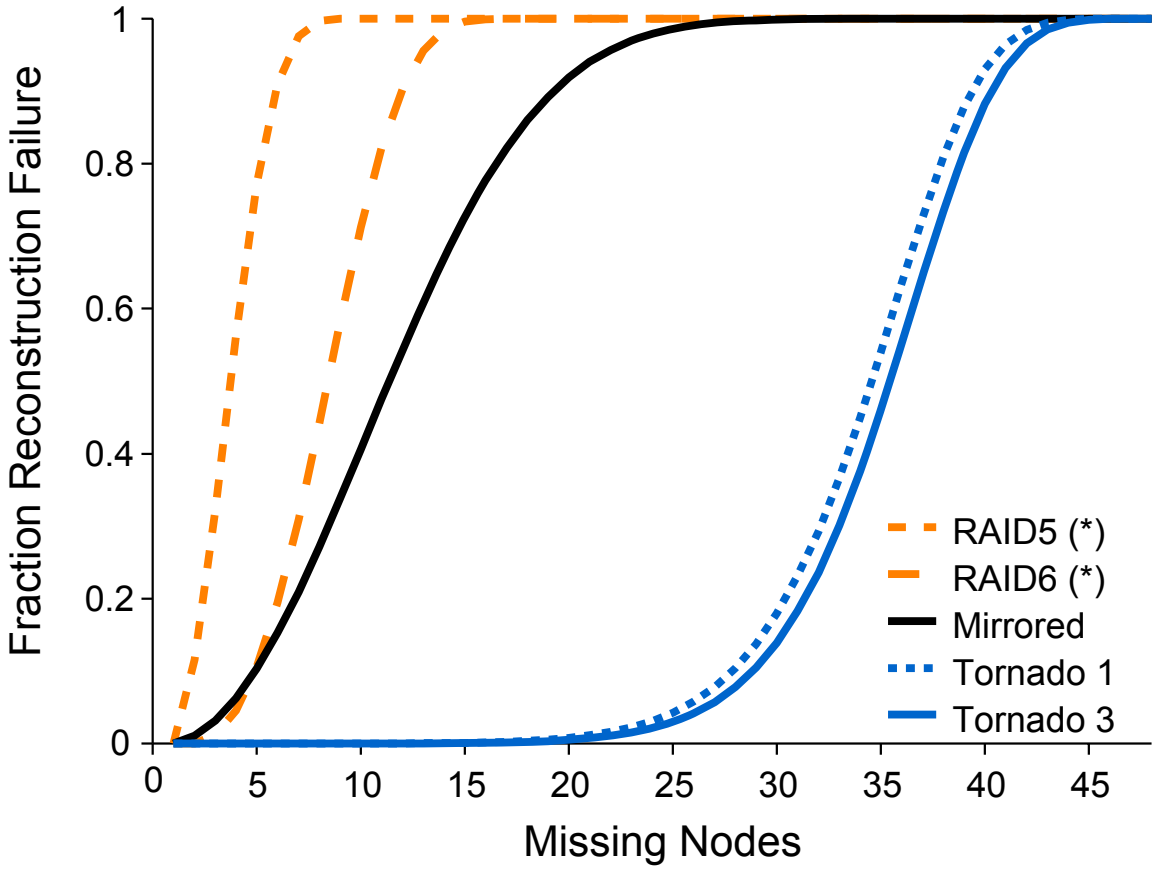
Comparison Single-Site Disk Configurations

- Compared to familiar RAID systems (8 drawers)
- All configurations have 96 devices
  - RAID 5
  - RAID 6
  - Mirrored (RAID 10)
  - Tornado Code graphs



- Results describe three specific Tornado Code graphs ("Tornado 1", "Tornado 2", "Tornado 3")

96-Device Tornado Code Fault Tolerance



	First Failure
RAID5	2
RAID6	3
Mirrored	2
Tornado 1	5
Tornado 3	5

- Tornado codes demonstrate better fault tolerance than mirroring with the same capacity overhead

Calculating Fault Tolerance

- The probability of any independent drive being lost is given by the binomial distribution:

$$P(k \text{ drives lost}) = (p)^k (1-p)^{n-k} \binom{n}{k}$$

- Because the experimental results are ***independent conditional*** probabilities, they may be summed:

$$P(\text{fail}) = \sum_{k=0}^{n \text{ drives}} P(\text{fail} \mid k \text{ drives lost}) P(k \text{ drives lost})$$

- This is a worst case fault tolerance calculation: it ignores all repairs

P(n nodes offline)

Device AFR:
p = 0.01

n offline	P(n offline)
0	0.38105
1	0.36950
2	0.17729
3	0.05611
4	0.01318
5	0.00245
6	0.00038
7	4.873E-05
8	5.476E-06
9	5.408E-07
10	4.753E-08
95	9.504E-189
96	1.000E-192

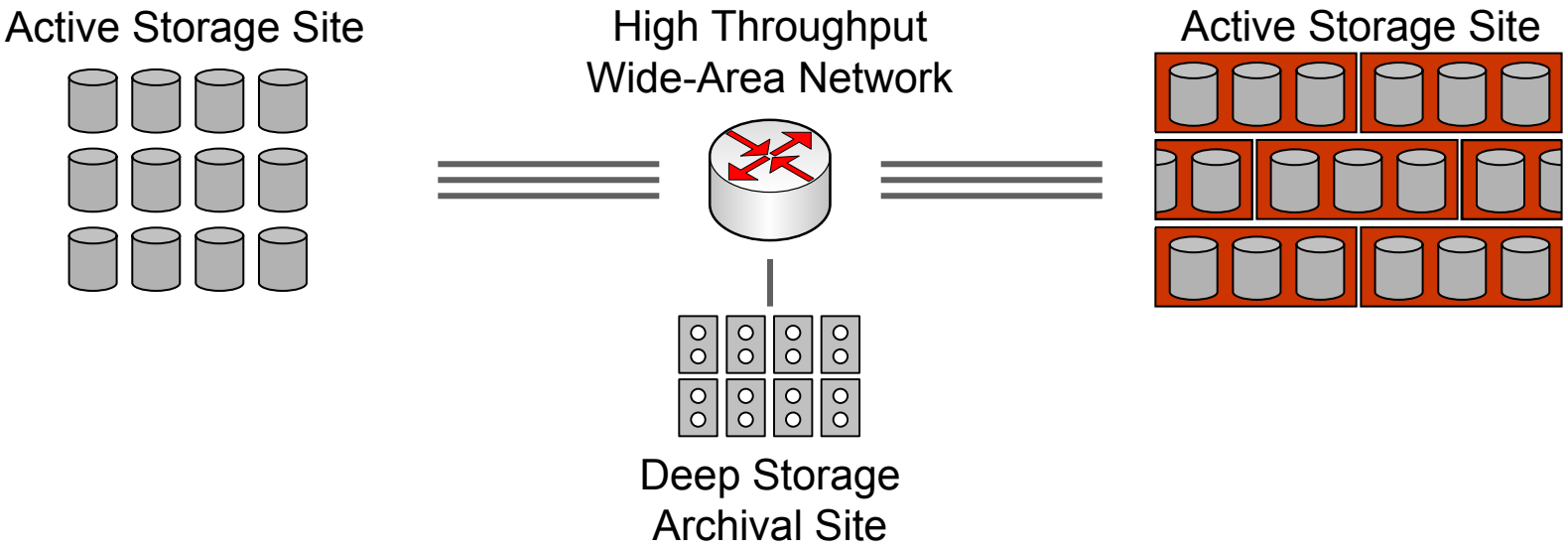
Single Site Stripe Fault Tolerance

Method	Data	Parity	P(fail)
Individual Disk			0.01000
Striping	96	0	0.61895
RAID5	88	8	0.04834
RAID6	80	16	0.00164
Mirrored	48	48	0.00479
Tornado 1	48	48	1.345E-09
Tornado 2	48	48	5.947E-10
Tornado 3	48	48	5.857E-10

Tornado Codes for Archival Storage

- Single site fault tolerance
- **Federated system fault tolerance**
- Optimization for MAID systems

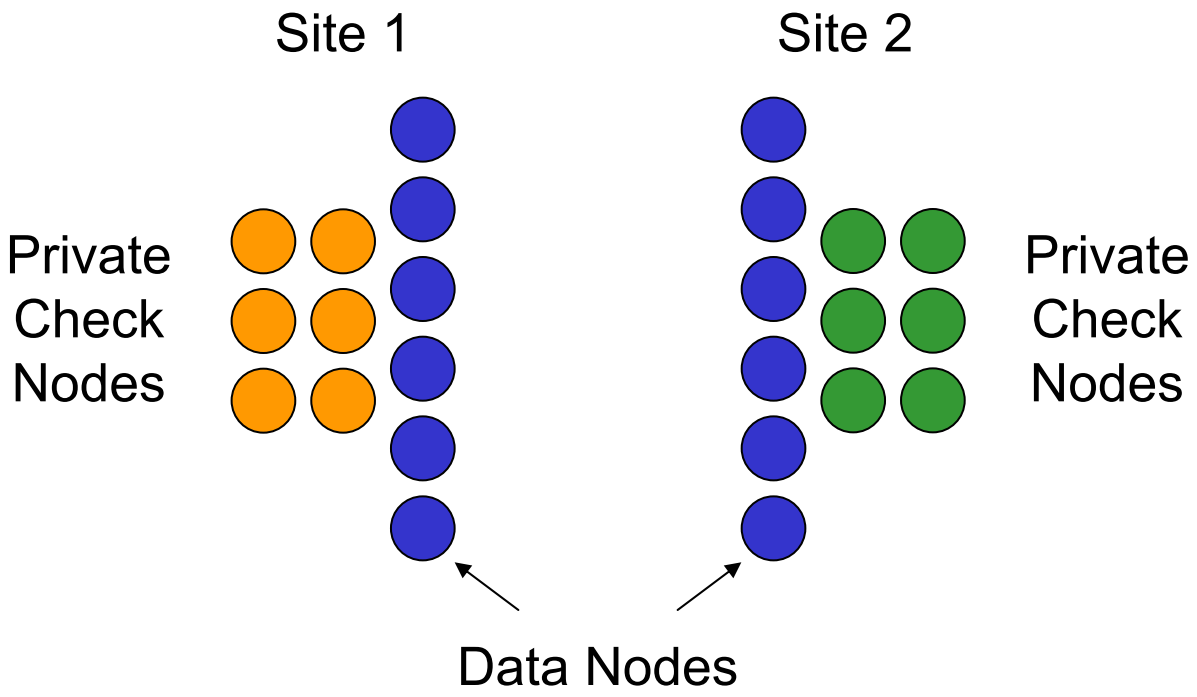
Distributed and Federated Storage



- Federated Storage
  - Distributed sites maintain subsets or complete copies
  - Replica selection and placement important
  - Data locality (close to researchers)
  - Data permanence (stewarding of historical data)

Multi-graph Federated Storage

- Federated storage with Tornado Codes
  - Each site uses a Tornado Code storage scheme
  - Different Tornado Code graphs used on each site
  - Sites may exchange data blocks to aid in reconstruction



Multi-graph Federated Storage

System	First Failure Detected (guided search)
Mirrored (4 copies)	4
Tornado 1 + Tornado 1	10
Tornado 1 + Tornado 2	17
Tornado 1 + Tornado 3	17
Tornado 2 + Tornado 3	19

- Cooperative graphs greatly enhance first failure
- Not all sites need to use Tornado Codes, just the ability to exchange missing data nodes

Federated Storage Applications

- Federated Tornado Code storage systems can reconstruct data even if all sites cannot do so independently
- Applications
  - Data stewarding
  - Collaborative storage and data grids
- Fits into existing technologies
  - Grid Bricks, MAID, tape systems
  - Data Grid interfaces (SRB, SRM/DataMover, GridFTP)

Tornado Codes for Archival Storage

- Single site fault tolerance
- Federated system fault tolerance
- **Optimization for MAID systems**

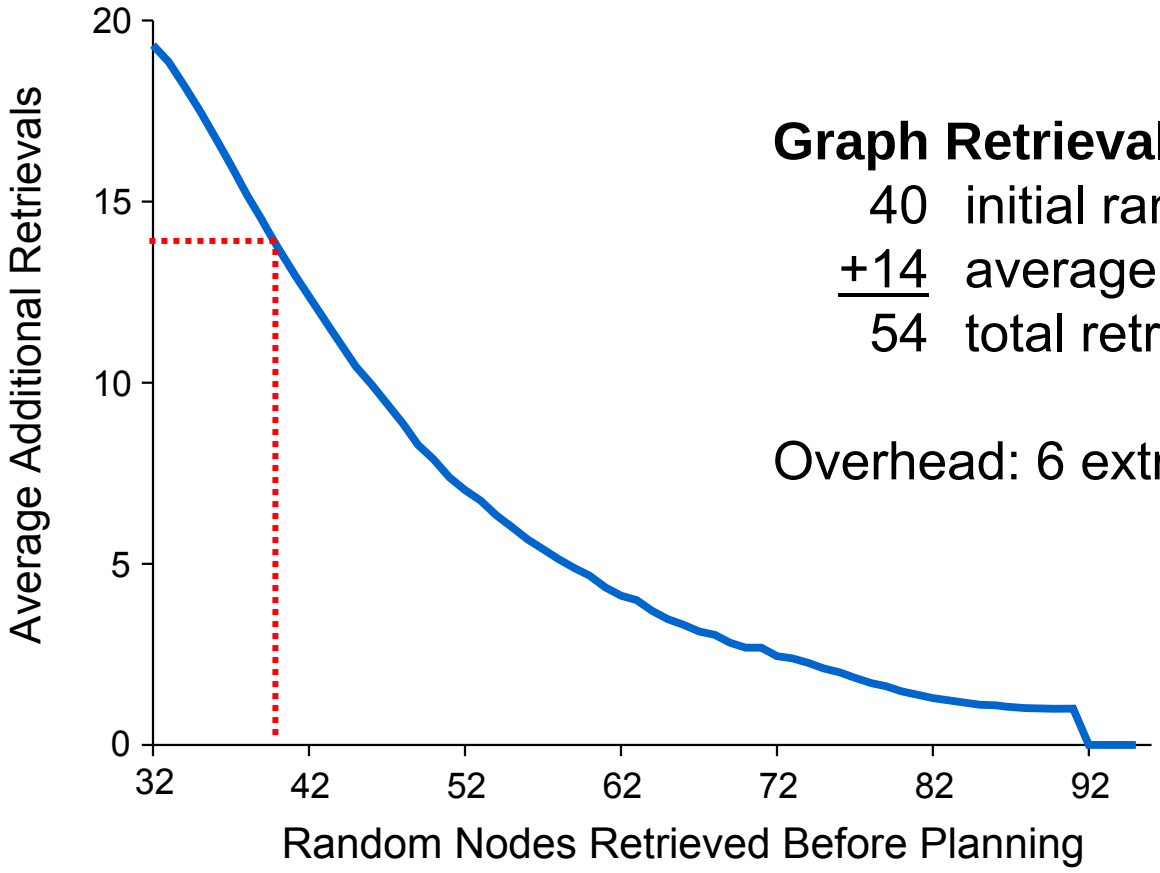
MAID Storage – Between Disk and Tape

- Massive Arrays of Idle Disks (MAID)
  - Disks stored with electrical power off
  - Devices activated by software to access data
  - Usually archival: write once, read occasionally
- Research work at CU Boulder by Grunwald, et. al.
- Recent product release from Copan Systems
  - ~1000 disks
  - ~10 second spin-up
  - “Highly unique RAID” supporting power management
- Tornado Codes provide high reliability and work well in archival storage applications

Tornado Codes and MAID Storage

- MAID objectives
  - Data permanence
  - Power conservation
- Conserve power by limiting online devices
  - Leverage redundant data
  - Selective device activation
- Device selection policies and optimizations
  1. Already online devices
  2. Random selection (collectively useful but prioritized)
  3. **Graph retrieval planning**

Graph Retrieval Planning Example



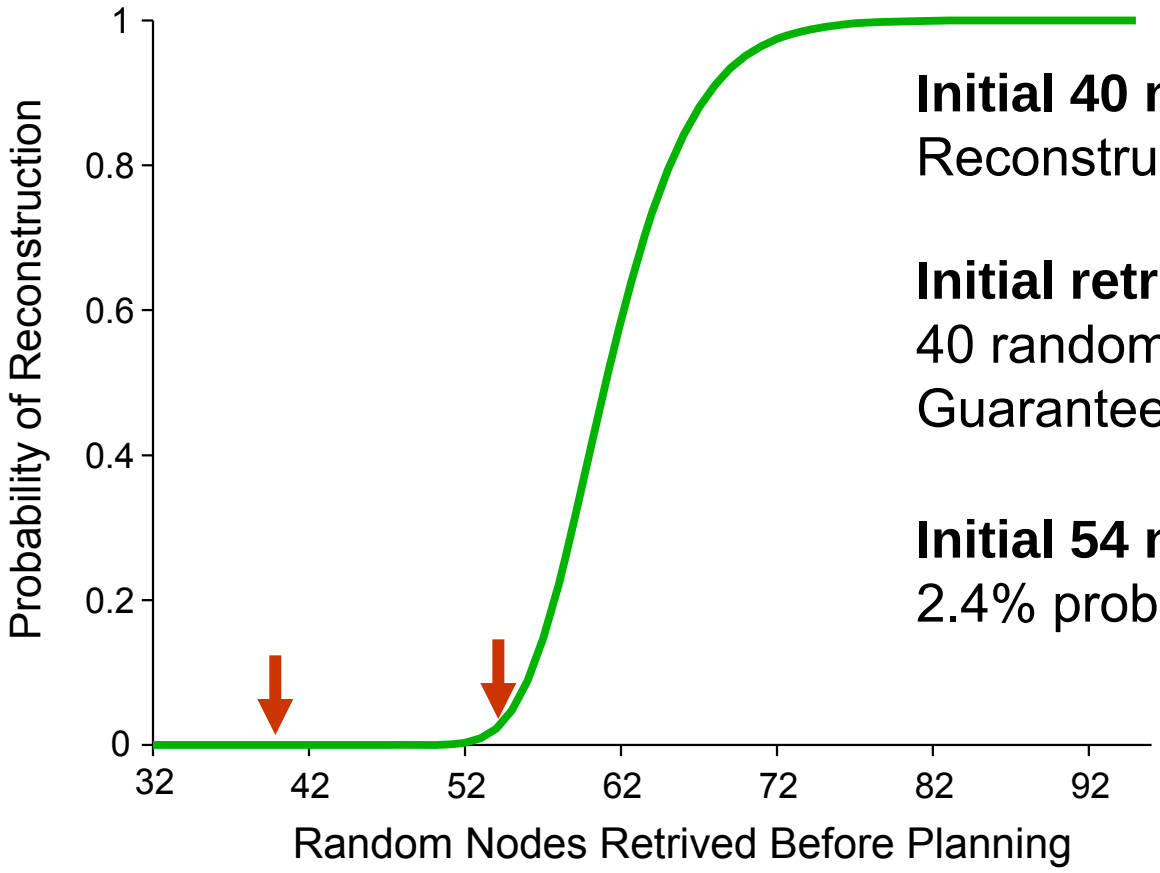
Graph Retrieval Plan example

40	initial random retrievals
+14	average additional retrievals
54	total retrievals

Overhead: 6 extra nodes (1.125)

- Planning takes time (5s for 40 prior retrievals) but can be performed in parallel to the first stage retrievals

Graph Retrieval Planning Example



Initial 40 node random retrieval
Reconstruction not possible

Initial retrieval + planning
40 random + 14 planned = 54
Guaranteed reconstruction

Initial 54 node random retrieval
2.4% probability of reconstruction

- Graph retrieval planning can reduce the number of device mounts required to retrieve data

Current Work

- Stripe retrieval simulations
  - Fixed size 960-disk system
  - Arbitrary retrieval requests
- Parameters
  - Number of stripes used for planning
  - Number of stripes passively retrieving
  - Percentage of disks online
- Metrics
  - Stripe retrieval time
  - Device mount count

Conclusions and Future Work

- Quantified “probabilistically successful” reconstruction for real Tornado Code graphs
  - First failure important
  - Verify then trust
  - Useful for distributed and federated storage systems
- Framework supports alternates to Tornado Codes
  - MIT Lincoln Erasure Codes (Cooley, 2003)
  - Plank’s small LDPC codes (2005, 2006)
  - … and other related LDPC-based codes

Future Work

- Emulated Filesystem
  - Stores encoded files
  - Simulates MAID behavior for evaluation
  - Works on real disks for data storage
- Goal: Reliable archival storage in a data grid
  1. Fault tolerance of real Tornado Code graphs
  2. Optimizations for efficient data reconstruction
  3. Operate within existing Data Grid standards
- Construct a working federated archival filesystem using Tornado Codes

Acknowledgements

- Thanks To:
  - Jason Cope
  - Dirk Grunwald
  - Sean McCreary
  - Michael Oberg

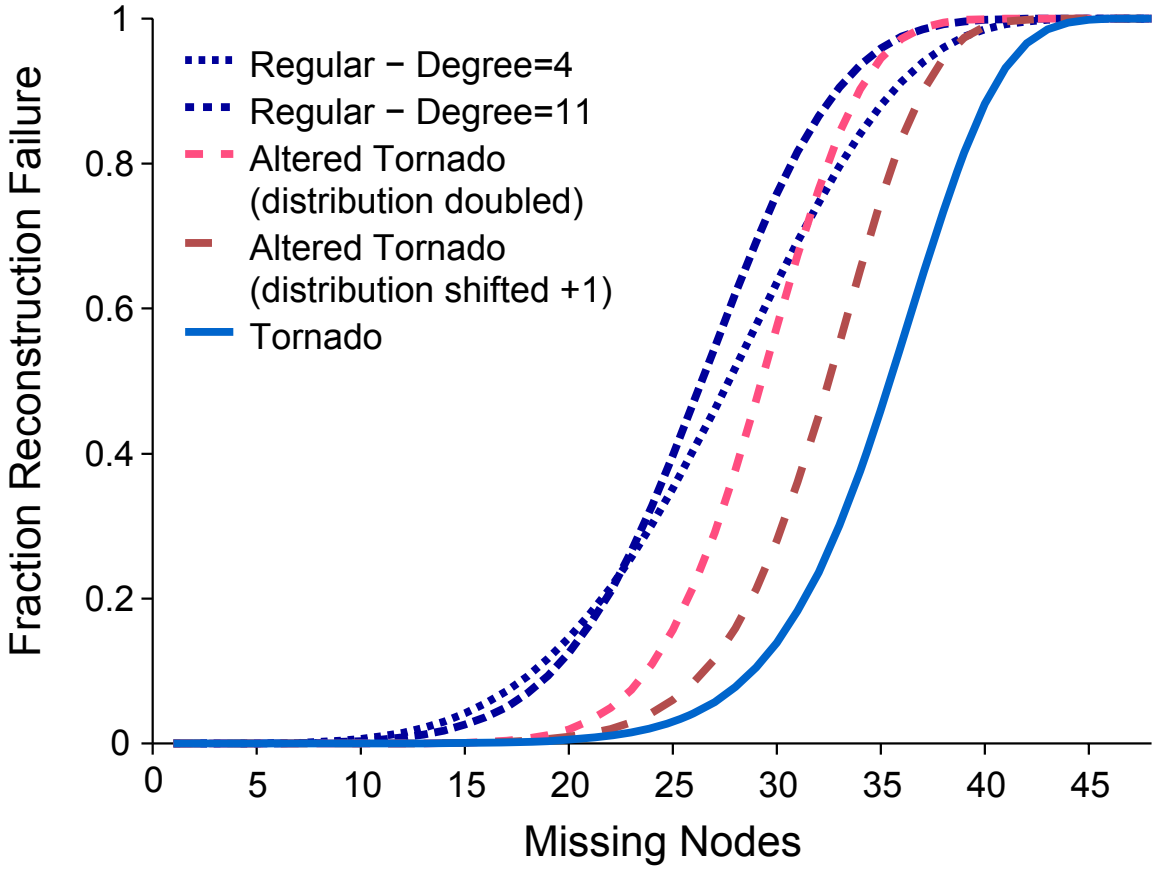
- Computer time was provided on equipment from:
  - NSF ARI Grant #CDA-9601817
  - NASA AIST grant #NAG2-1646
  - DOE SciDAC grant #DE-FG02-04ER63870
  - NSF sponsorship of the National Center for Atmospheric Research
  - IBM Shared University Research (SUR) program

Questions?

Matthew Woitaszek
matthew@ucar.edu

NCAR

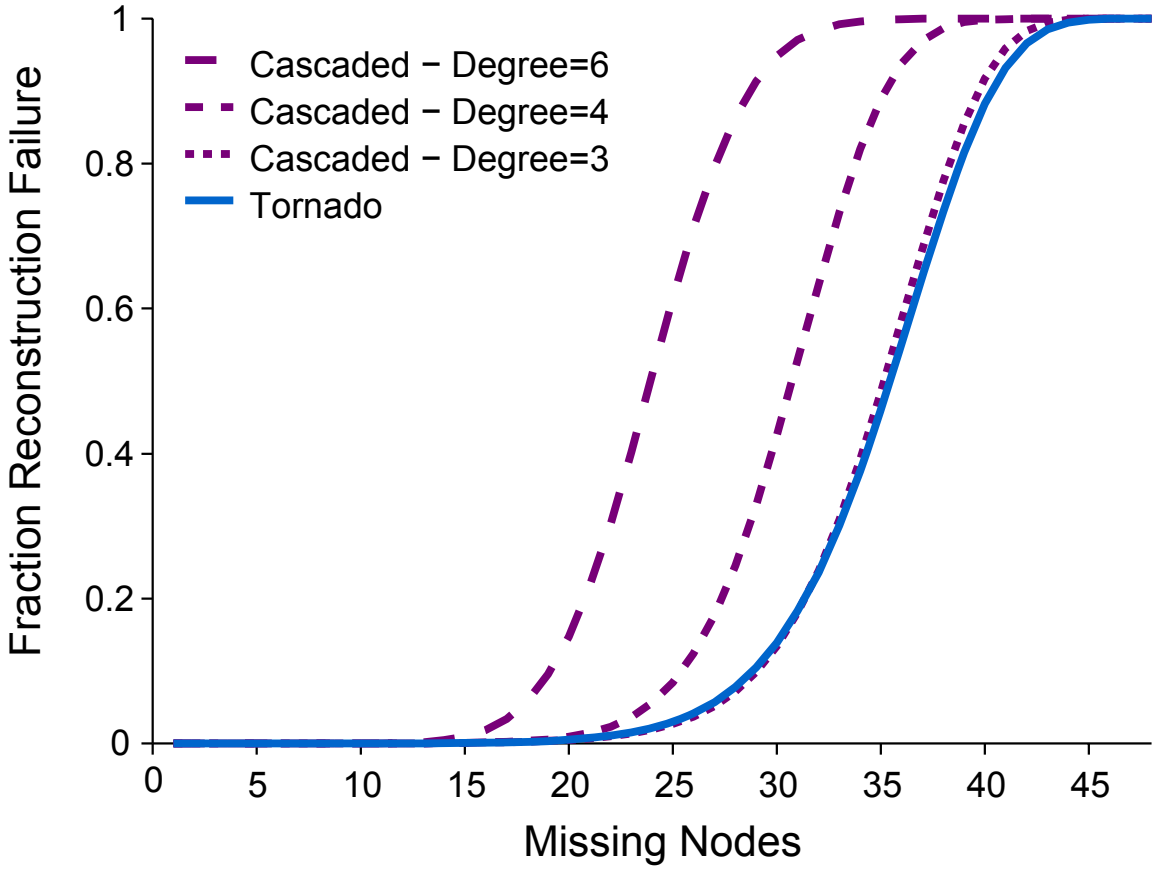
96-Device Tornado vs. Alternate Fault Tolerance



	First Failure
Regular 4	4
Regular 11	4
Doubled	5
Shifted	5
Tornado	5

- Tornado Code theory shows better results than token adjustments or simple regular bipartite graphs

96-Device Tornado vs. Alternate Fault Tolerance



	First Failure
Degree 6	5
Degree 4	4
Degree 3	4
Tornado	5

- Cascaded fixed degree graphs with random edge permutations
- Curve shape matches but first failure is important characteristic

Fault Tolerance I

P(fail | n nodes offline)

Experimentally calculated results

n Offline	RAID5	RAID6	Striping	Mirroring	g096-test02	Tornado 1 g096-05-03	Tornado 2 g096-06-03	Tornado 3 g096-tc05a
0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0
2	0.115789	0	1	0.010526	0.000219	0	0	0
3	0.322732	0.012318	1	0.031579	0.000665	0	0	0
4	0.562957	0.045718	1	0.062845	0.001344	0	0	0
5	0.771965	0.105712	1	0.103540	0.002263	0.000000	0.000000	0.000000
6	0.909780	0.194707	1	0.152613	0.003440	0.000001	0.000000	0.000001
7	0.975944	0.309697	1	0.209144	0.004884	0.000005	0.000003	0.000002
8	0.996782	0.442486	1	0.271360	0.006577	0.000012	0.000007	0.000006
9	1	0.581352	1	0.337706	0.008584	0.000033	0.000018	0.000019
10	1	0.711812	1	0.406102	0.010938	0.000067	0.000036	0.000037
95	1	1	1	1	1	1	1	1
96	1	1	1	1	1	1	1	1

Fault Tolerance II

- The probability of any independent drive being lost is given by the binomial distribution:

$$P(k \text{ drives lost}) = (p)^k (1-p)^{n-k} \binom{n}{k}$$

- Because the experimental results are ***independent conditional*** probabilities, they may be summed:

$$P(\text{fail}) = \sum_{k=0}^{n \text{ drives}} P(\text{fail} \mid k \text{ drives lost}) P(k \text{ drives lost})$$

- This is a worst case fault tolerance calculation: it ignores all repairs.

P(n nodes offline)

p = 0.01

n Offline	P(n offline)
0	0.38105
1	0.36950
2	0.17729
3	0.05611
4	0.01318
5	0.00245
6	0.00038
7	4.873E-05
8	5.476E-06
9	5.408E-07
10	4.753E-08
95	9.504E-189
96	1.000E-192

1

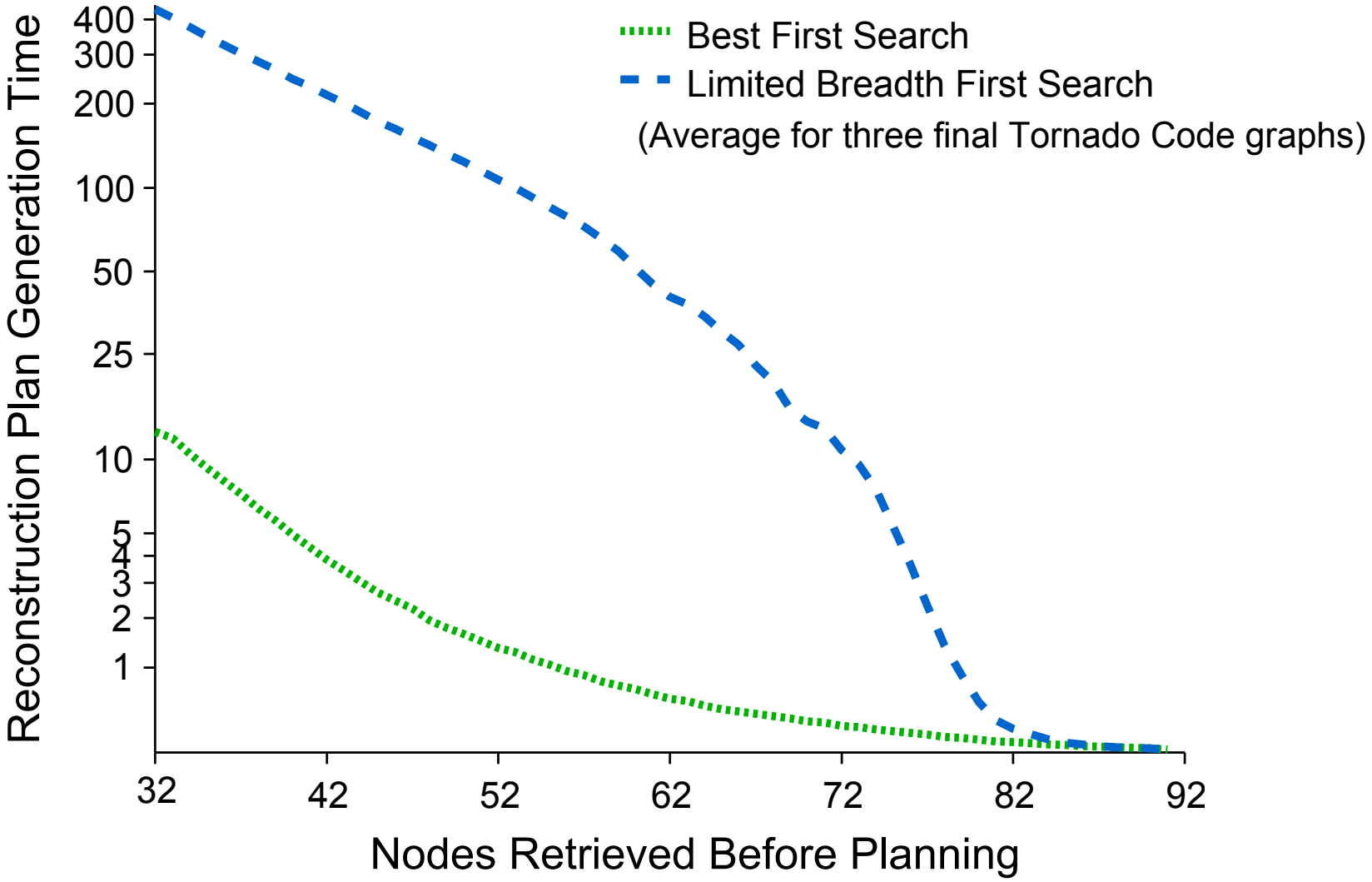
Fault Tolerance III

P(fail) = Sum[P(n nodes offline)*P(fail | n nodes offline)]

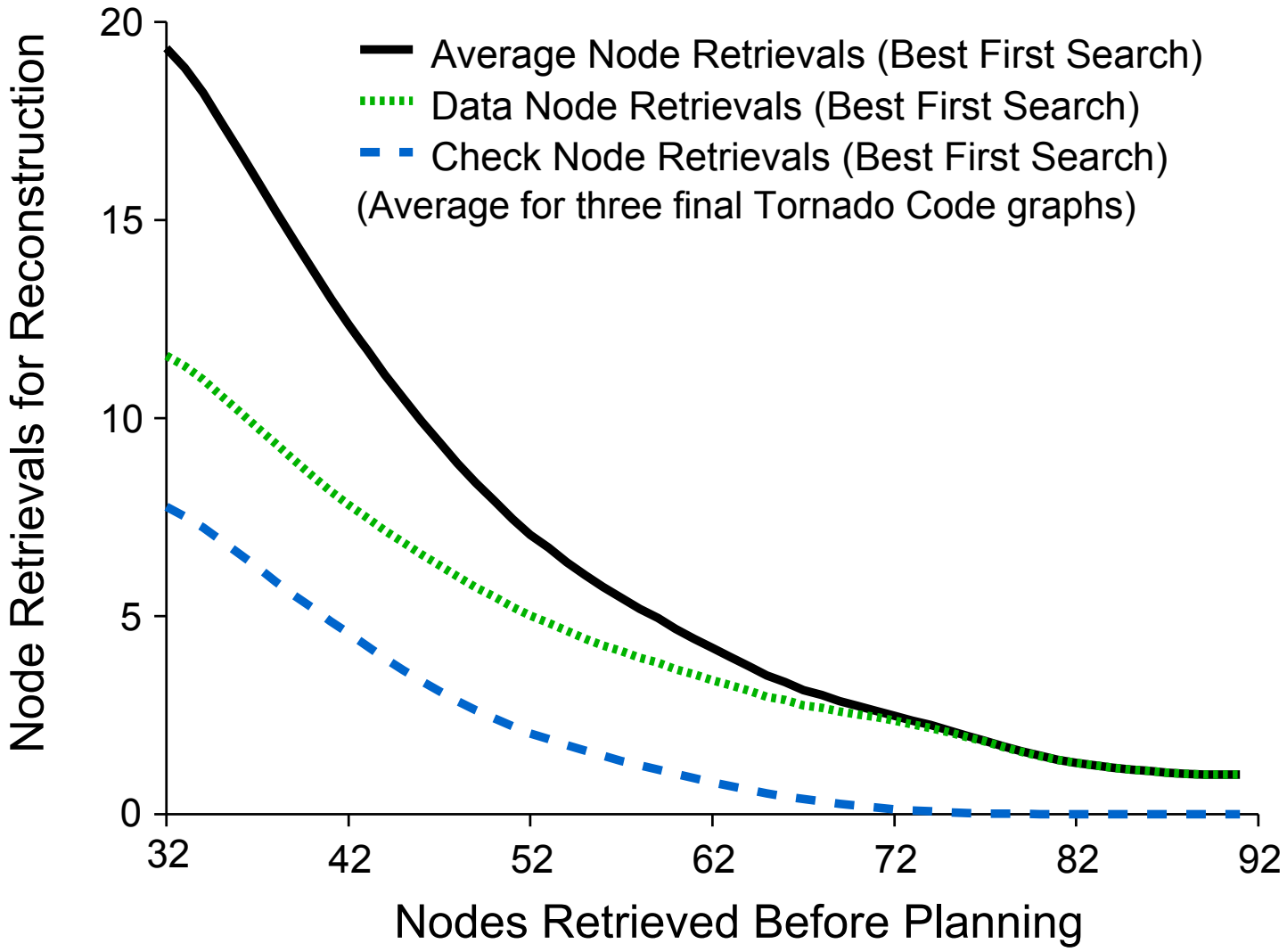
Independent conditional probabilities

n Offline	RAID5	RAID6	Striping	Mirroring	g096-test02	Tornado 1 g096-05-03	Tornado 2 g096-06-03	Tornado 3 g096-tc05a
0	0	0	0	0	0	0	0	0
1	0	0	0.3695002	0	0	0	0	0
2	2.053E-02	0.000E+00	1.773E-01	1.866E-03	3.888E-05	0	0	0
3	1.811E-02	6.912E-04	5.611E-02	1.772E-03	3.731E-05	0	0	0
4	7.418E-03	6.025E-04	1.318E-02	8.281E-04	1.771E-05	0	0	0
5	1.891E-03	2.589E-04	2.449E-03	2.536E-04	5.543E-06	4.791E-10	2.395E-10	2.395E-10
6	3.414E-04	7.306E-05	3.752E-04	5.726E-05	1.291E-06	5.466E-10	1.608E-10	1.929E-10
7	4.756E-05	1.509E-05	4.873E-05	1.019E-05	2.380E-07	2.304E-10	1.430E-10	1.073E-10
8	5.458E-06	2.423E-06	5.476E-06	1.486E-06	3.602E-08	6.743E-11	3.952E-11	3.393E-11
9	5.408E-07	3.144E-07	5.408E-07	1.826E-07	4.642E-09	1.764E-11	9.843E-12	1.001E-11
10	4.753E-08	3.383E-08	4.753E-08	1.930E-08	5.198E-10	3.168E-12	1.705E-12	1.757E-12
95	9.504E-189	9.504E-189	9.504E-189	9.504E-189	9.504E-189	9.504E-189	9.504E-189	9.504E-189
96	1.000E-192	1.000E-192	1.000E-192	1.000E-192	1.000E-192	1.000E-192	1.000E-192	1.000E-192
	0.04834	0.00164	0.61895	0.00479	0.00010	1.34E-09	5.95E-10	5.86E-10

Graph Retrieval Plan Generation Time



Additional Retrievals by Initial Retrieval



Additional Retrievals by Data Nodes Missing