



# Scheduling for Improved Write Performance in a Cost-Effective, Fault-Tolerant Parallel Virtual File System (CEFT-PVFS)

Yifeng Zhu, Hong Jiang,  
Xiao Qin, Dan Feng, David Swanson  
Department of Computer Science and Engineering  
University of Nebraska – Lincoln  
June, 2003

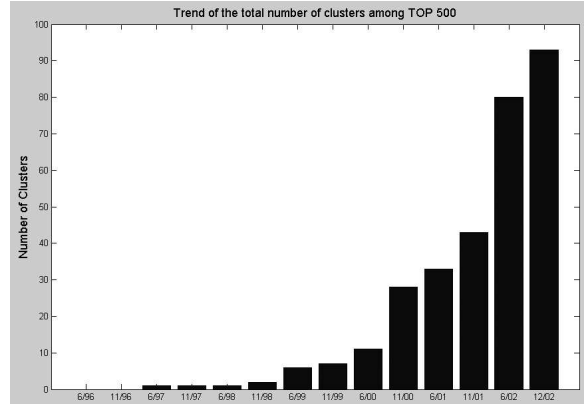


## Outline

- Motivations of CEFT-PVFS
  - I/O bottleneck in cluster computing
  - Storage of high reliability
- Overview of CEFT-PVFS
- Previous works
  - Reliability Improvement
  - Read Performance Improvement
- Write performance improvement
  - Examine the simplest configuration
  - Heuristic rules
  - Evaluations
- Conclusions and future work



# Cluster Computing



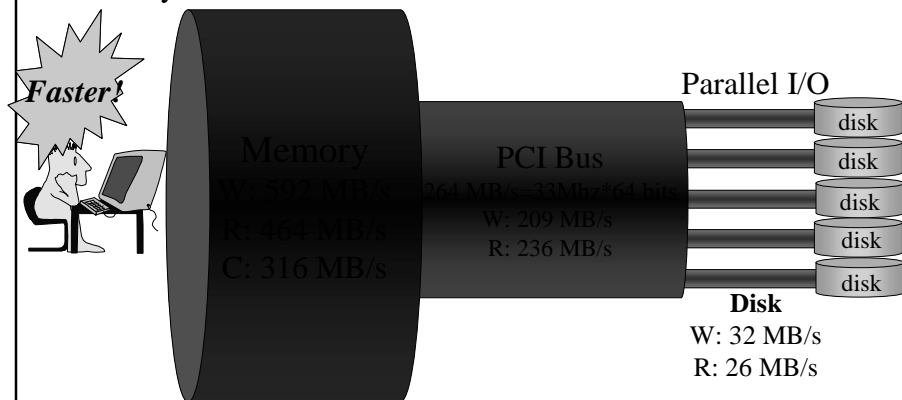
- Cluster computing is the fastest growing platform.
- 93 clusters are listed among the Top 500 today.
- The power of cluster computing tends to be bounded by I/O performance.

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions



# Parallel I/O

The following data is measured on the PrairieFire Cluster at University of Nebraska - Lincoln:



Parallel I/O is to alleviate I/O bottleneck in clusters.

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions



## PVFS: Reliability Issue

- PVFS is RAID-0 (striping) style, without any redundancy.
- PVFS has data reliability problem.

Assume: ( 1 ) MTTF of a disk is 5 years;

( 2 ) Other components are fault-free;

MTTF of PVFS = MTTF of 1 node  $\div$  N

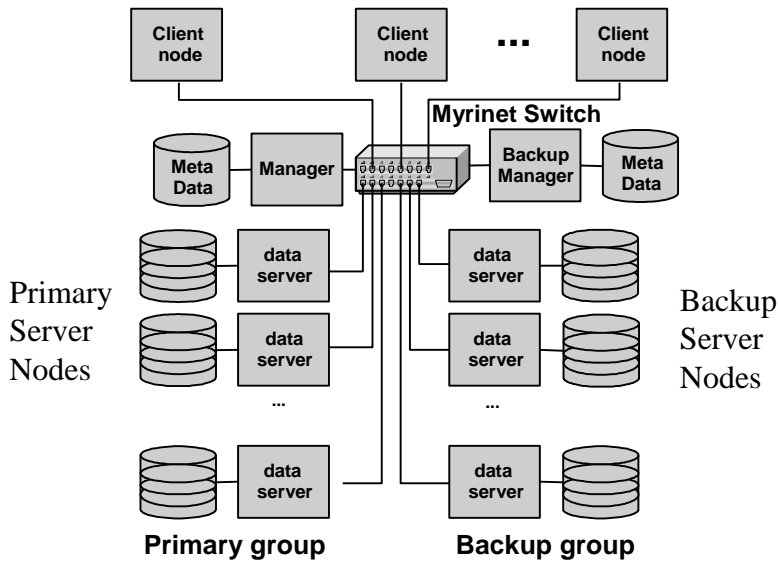
= 5 years  $\div$  128 = 14 days.

Like disk arrays, PVFS is too unreliable to be useful.

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions



## Diagram of CEFT-PVFS

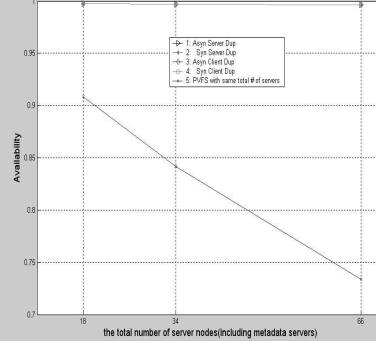
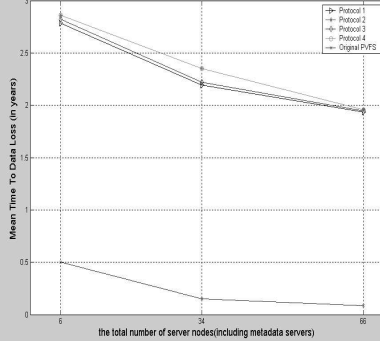


• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions



# Reliability & Availability Comparisons

Reliability comparison of CEFT-PVFS and original PVFS



## Reliability Comparisons

## Availability Comparisons

Mirroring can improve the reliability by a factor of over 40 (4000%).

Y. Zhu, "CEFT-PVFS: A Cost-Effective, Fault-Tolerant parallel virtual file system", Master Thesis, Department of Computer Science and Engineering, University of Nebraska – Lincoln, Dec. 2002.

- Motivations
- Overview
- Previous Work
- Benchmark
- Algorithm
- Performance
- Conclusions



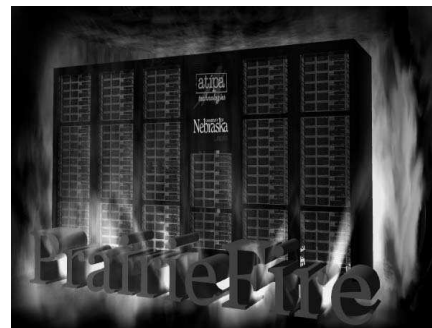
# System Environments

## PrairieFire Cluster

- 107<sup>th</sup> fastest in the world ( peak: 716 Gflops )
- Equipped with 128 IDEs, 2.6 TeraBytes totally
- 128 nodes, 256 processors
- Myrinet and Fast Ethernet
- Linux 2.4 and GM-1.5.1

## Performance:

- TCP/IP: 126.5 MB/s
- Disk writes: 32 MB/s
- Disk reads: 26 MB/s



- Motivations
- Overview
- Previous Work
- Benchmark
- Algorithm
- Performance
- Conclusions



# Performance Benchmark

- Each client reads/writes a disjointed portion of an existing file concurrently;
- Aggregate performance = Total data size ÷ Average Response Time;
- Performance per server = Aggregate performance ÷ Number of servers;

Pseudocode of the benchmark

**For all clients:**

```

synchronize all clients using MPI barrier;
t1 = current time;
open a file;
synchronize all clients using MPI barrier;
loop to write data;
close the file;
t2 = current time;
ct = t1 - t2; /* overall completion time */
send ct to client 0;

```

**For client 0:**

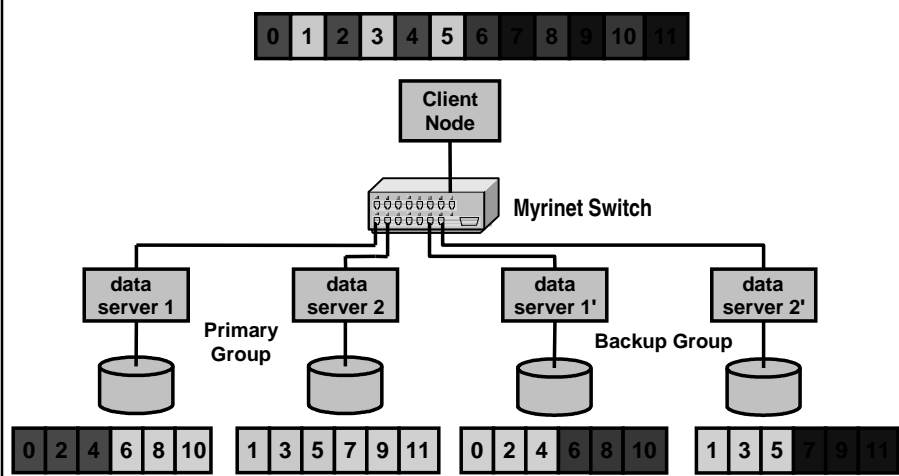
```

find maximum of ct; /*find slowest client */
calculate write throughput using maximum ct;

```



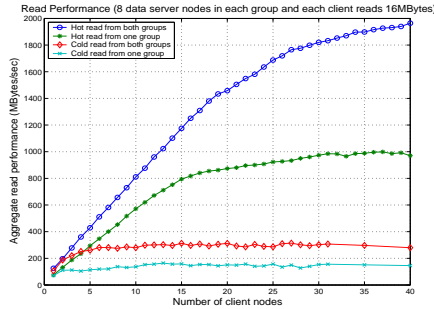
# Double Degree of Parallelism for Read Operations



Read interleaved data from both groups simultaneously to boost the peak read performance.



## Read Performance: All clients read the same amount data but the total number of clients changes



- 8 data servers in one group and each client read 16 MB

- Conclusion: Peak read performance is doubled in both cold and hot read.

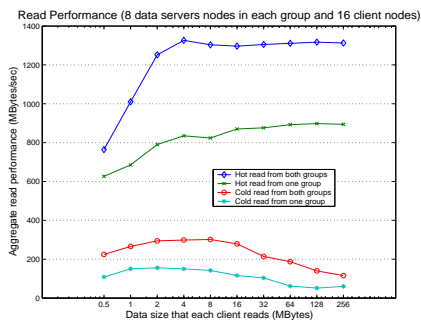
Peak Read Performance (MB/s)

	Aggregate	Per server
Hot read from both groups	1964.4	122.8
Hot read from one group	998.5	124.8
Cold read from both groups	313.7	19.6
Cold read from one group	164.3	20.5

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions



## Read Performance: The total number of clients is fixed while the data size changes



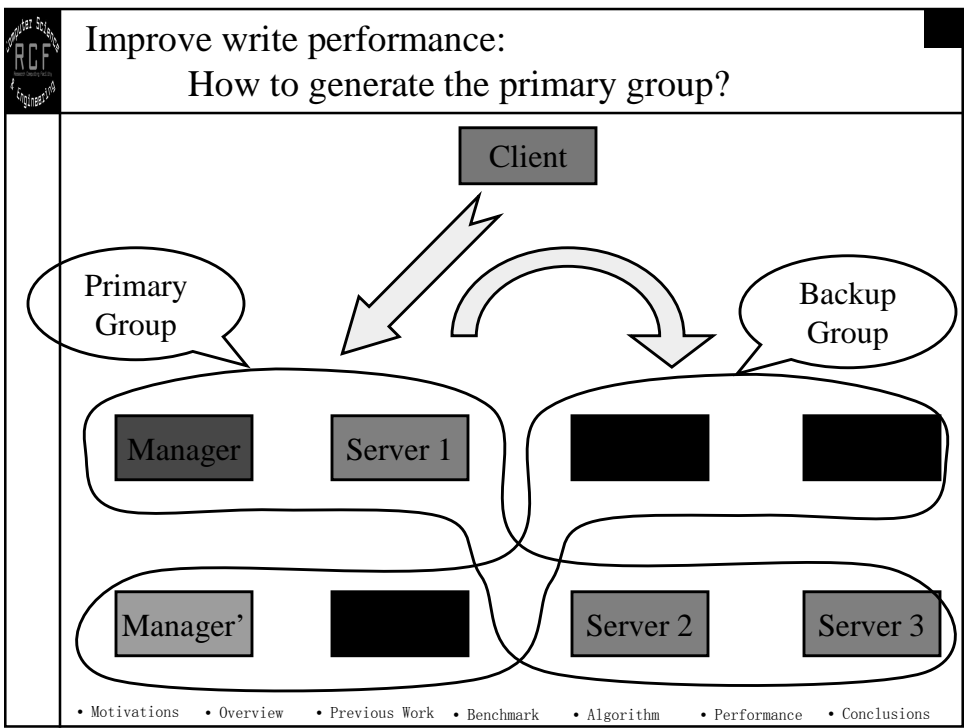
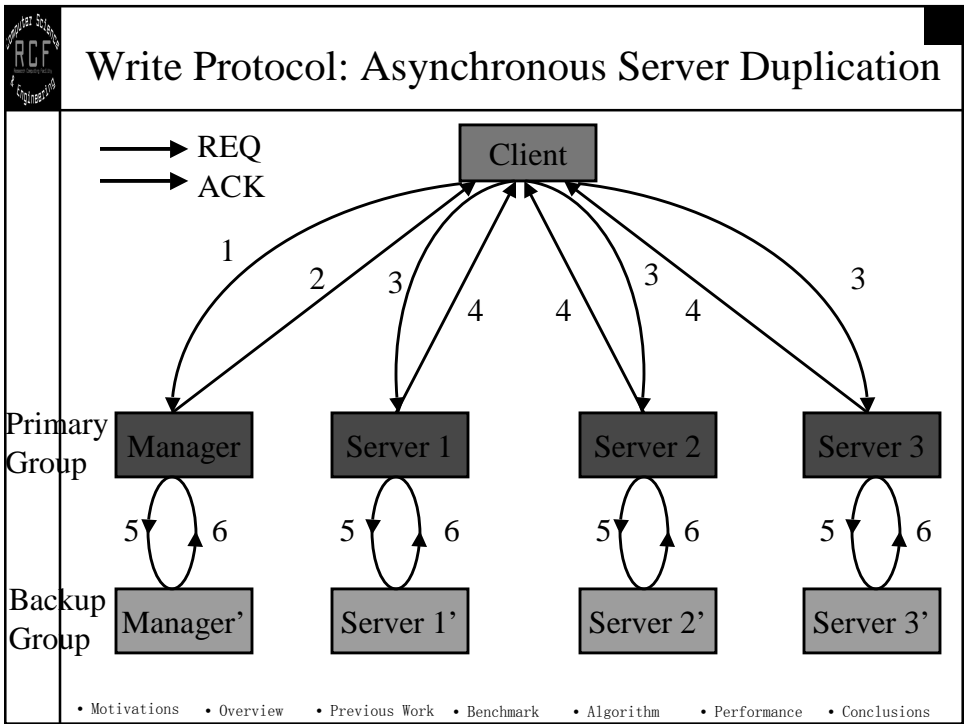
- 8 data servers in one group and 16 clients

- Conclusion: In the range of tests, the proposed method improves the hot and cold read performance 69% and 91% on the average respectively.

Peak Read Performance (MB/s)

	Aggregate	Per server
Hot read from both groups	1326.3	82.9
Hot read from one group	897.9	112.2
Cold read from both groups	316.8	19.8
Cold read from one group	160.8	20.1

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions





## Write Performance Improvement

Examine the dynamic behavior of individual server:

1. Simplest configuration: both groups contain only one data server and one metadata server
2. Simplest I/O access pattern, in which only one client writes a new file to the data server.

Under different system resources load:

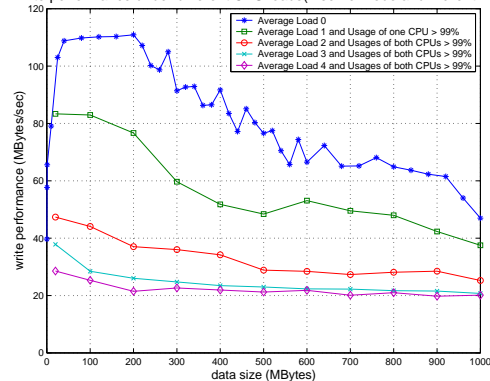
1. Write performance under different network traffic disturbance
2. Write performance when the memory and disk are heavily loaded

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions



## Influence of CPU on Write Performance

Write performance under different CPU loads(1 server node and 1 client node)

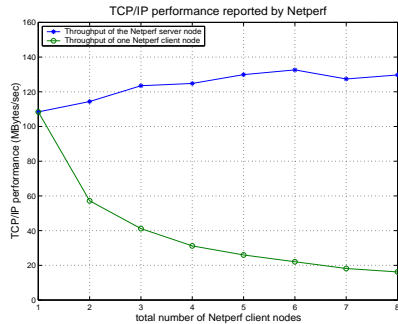


1. CPU is not the bottleneck if both CPUs are not 99% utilized.
2. Write performance will be reduced by approximately 31%, 60%, 70%, and 73% on average if the utilization of both CPUs is 99% and the average load is 1, 2, 3, and 4, respectively.

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions



## TCP/IP Characteristics on clusters



**Multiple Netperf clients communicate with one Netperf server simultaneously.**

**Server average throughput: 126.51 MB/s, shared by all the clients.**

**Server average CPU utilization: 47%.**

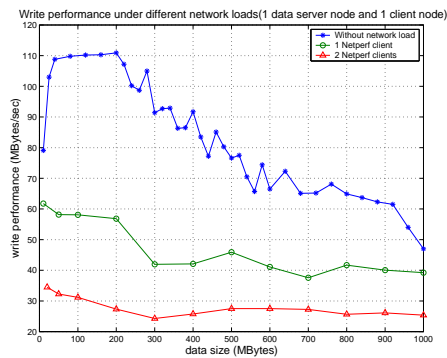
**PCI bus read: 236 MB/sec and write: 209 MB/sec**

1. The bottleneck of the TCP/IP performance is likely located at the TCP/IP stack at the server side.
2. The bottleneck of I/O operations could potentially shift from disks to the TCP/IP stack.

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions



## Write performance under different network traffic disturbance

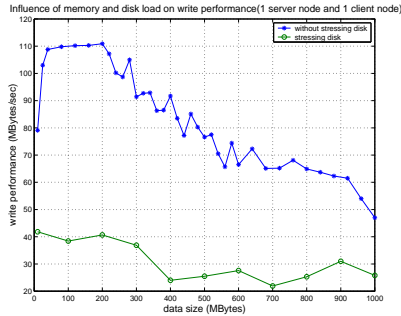


1. Place data server and Netperf server on the same node.
2. Netperf and data server daemon almost evenly share the TCP/IP bandwidth.
3. When data size is large, performance degrades due to the negative impact of memory shortage.

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions



## Write performance when the memory and disk are heavily loaded



### Program to stress the memory and disk

```

1. M = allocate(1 MBytes);
2. create a file named F;
3. while(1)
4.   if(size(F) > 1 GB)
5.     truncate F to zero byte
6.   else
7.     synchronously append the data
8.     in M to the end of F.
9. end of while.

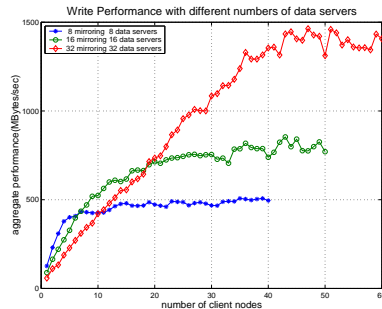
```

1. When the stress program is running, both CPUs are 95% idle.
2. The disk bandwidth is nearly equally shared by all the I/O-intensive processes.

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions

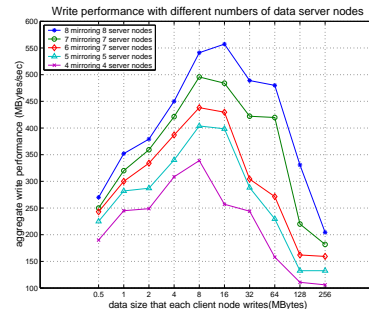


## Skipping a node while striping?



Aggregate write performance of CEFT-PVFS when each client writes 16 MBytes data to the servers without stressing

1. A larger number of server nodes does not necessarily result in a proportionally higher write performance.
2. Without stressing, skipping can potentially improve performance.
3. When hot-spot exists, skipping can avoid or reduce degradation.



Aggregate write performance of CEFT-PVFS when the total number of clients nodes is 16 without stressing

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions



# Scheduling algorithms

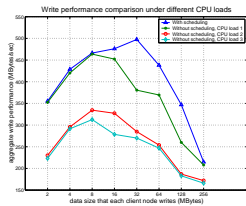
Algorithm:

1. if  $\min(\text{CPU}_{i1}, \text{CPU}_{i2}, \text{CPU}_{j1}, \text{CPU}_{j2}) \geq 99\%$  and  $\min(\text{LOAD}_i, \text{LOAD}_j) > 2$ 
  - 2. set the skipping flag;
3. else if  $\min(\text{CPU}_{i1}, \text{CPU}_{i2}) \geq 99\%$  and  $\text{LOAD}_i > 2$ 
  - 4. choose node  $j$ ;
5. else if  $\min(\text{CPU}_{i1}, \text{CPU}_{j2}) \geq 99\%$  and  $\text{LOAD}_j > 2$ 
  - 6. choose node  $i$ ;
7. else if  $\text{MEM}_i > \text{FSIZE}$  and  $\text{MEM}_j > \text{FSIZE}$ 
  - 8. if  $\text{NET}_i \leq 0.5 * \text{DISK}_{\max}$  and  $\text{NET}_j \leq 0.5 * \text{DISK}_{\max}$ 
    - 9. set the skipping flag;
  - 10. else
    - 11. choose  $i$  if  $\text{NET}_i \geq \text{NET}_j$ ; otherwise, choose  $j$ ;
    - 12. end
13. else if  $\text{MEM}_i \leq \text{FSIZE}$  and  $\text{MEM}_j \leq \text{FSIZE}$ 
  - 14. if  $\min(\text{DISK}_i, \text{NET}_i) \leq 0.5 * \text{DISK}_{\max}$ 
    - 15. and  $\min(\text{DISK}_j, \text{NET}_j) \leq 0.5 * \text{DISK}_{\max}$ 
      - 16. set the skipping flag;
    - 17. else
      - 18. choose  $i$  if  $\min(\text{DISK}_i, \text{NET}_i) > \min(\text{DISK}_j, \text{NET}_j)$ ;
      - 19. otherwise  $j$ ;
      - 20. end
  - 21. else
    - 22. choose  $i$  if  $\text{MEM}_i > \text{FSIZE} > \text{MEM}_j$  and  $\text{NET}_i \geq 0.5 * \text{DISK}_{\max}$
    - 23. choose  $j$  if  $\text{MEM}_j > \text{FSIZE} > \text{MEM}_i$  and  $\text{NET}_j \geq 0.5 * \text{DISK}_{\max}$
    - 24. choose  $i$  if  $\text{MEM}_i > \text{FSIZE} > \text{MEM}_j$  and  $\text{NET}_i \geq 0.5 * \text{DISK}_{\max}$
    - 25. choose  $j$  if  $\text{MEM}_j > \text{FSIZE} > \text{MEM}_i$  and  $\text{NET}_j \geq 0.5 * \text{DISK}_{\max}$
    - 26. otherwise set the skipping flag;
  - 27. end

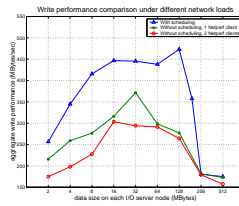
- Motivations
- Overview
- Previous Work
- Benchmark
- Algorithm
- Performance
- Conclusions



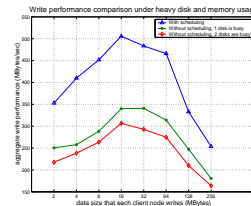
# Performance Evaluations



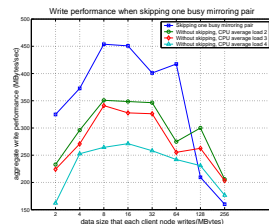
Stress CPUs on one server node



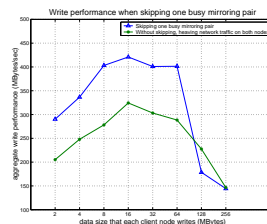
Stress network on one server



Stress disk and memory on one server node



Stress CPUs on one mirroring pair



Stress disk and memory on one mirroring pair

- Motivations
- Overview
- Previous Work
- Benchmark
- Algorithm
- Performance
- Conclusions



## Conclusion

- The I/O performance can be significantly reduced if the data servers, which also serve as computational nodes in a cluster, are heavily loaded by applications running in the cluster.
- Since in CEFT-PVFS each data item has two copies and they are stored in two different nodes, it provides an opportunity for scheduling algorithms to exploit the workload disparity between these two nodes in a mirroring pair or among all mirroring pairs within a striping group.
- A simple but effective algorithm is developed and its performance is evaluated in a CEFT-PVFS with 16 servers under different workload conditions. The performance measured shows that it significantly improves the overall I/O write performance when the system is under an unbalanced workload.

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions



## Future Work

- Evaluate the read performance in a more comprehensive and realistic benchmark;
- Reading data simultaneously only benefits large read. Improve the performance of small reads by caching.

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions



## References

1. Y. Zhu, "CEFT-PVFS: A cost-effective, fault-tolerant parallel virtual file system", Master Thesis, Department of Computer Science and Engineering, University of Nebraska – Lincoln, Dec. 2002.
2. Y. Zhu, H. Jiang, X. Qin, D. Feng, and D. Swanson, "Improved read performance in a Cost-Effective, Fault-Tolerant Parallel Virtual File System (CEFT-PVFS)," in Proceeding of 3<sup>rd</sup> IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID), 2003, p. 730-735
3. X. Qin, H. Jiang, Y. Zhu, and D. Swanson, "Dyamic load balancing for I/O- and Memory-Intensive workload in Clusters using a Feedback Control Machanism", the 9<sup>th</sup> international Euro-Par conference on parallel processing, Klagenfurt, Austria, Aug. 2003 (to appear)

• Motivations • Overview • Previous Work • Benchmark • Algorithm • Performance • Conclusions