High Performance Computing

Taichiro Suzuki
Tokyo Institute of Technology
Dept. of mathematical and computing sciences
Matsuoka Lab.

Review Paper

Two-Level Checkpoint/Restart Modeling for GPGPU

Supada Laosooksathit, Nichamon Naksinehaboon,

Chokchai Leangsuksan

Department of Computer Science, College of Engineering and Science Louisiana Tech University, Ruston 71272, USA

AICCSA 2011

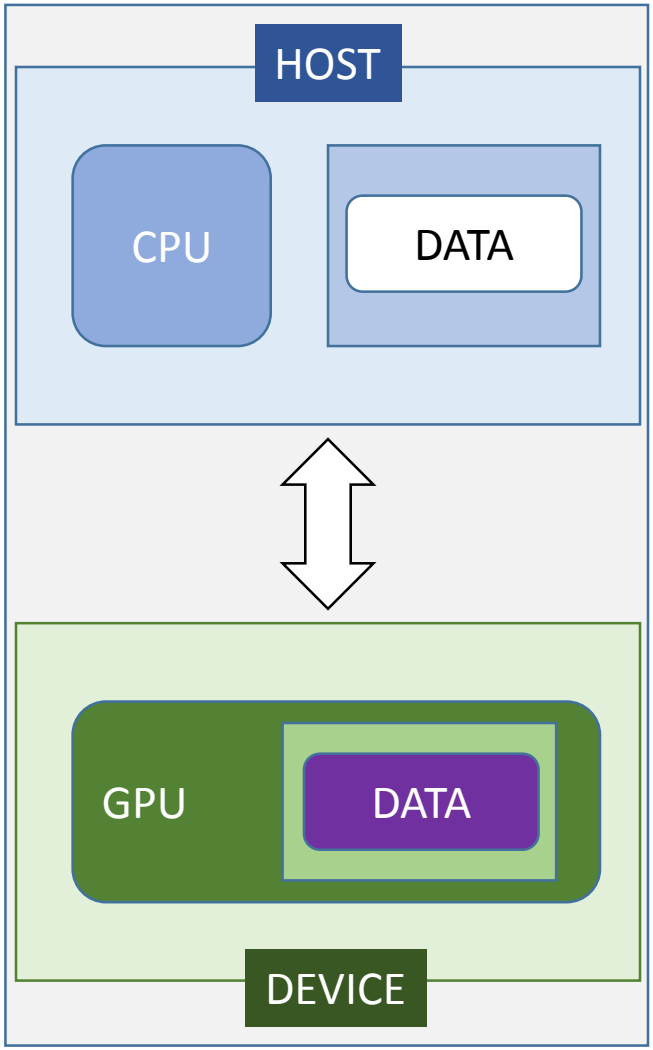
GPGPU

Nowadays, supercomputing applications has increasingly explored power of GPU and the cluster of GPUs for non-graphic applications [p276]

→**GPGPU**

- GPU works as a co-processor of CPU
- Flow of execution of GPU application

1. Memory copy from Host to Device
2. Kernel execution
3. Memory copy from Device to Host

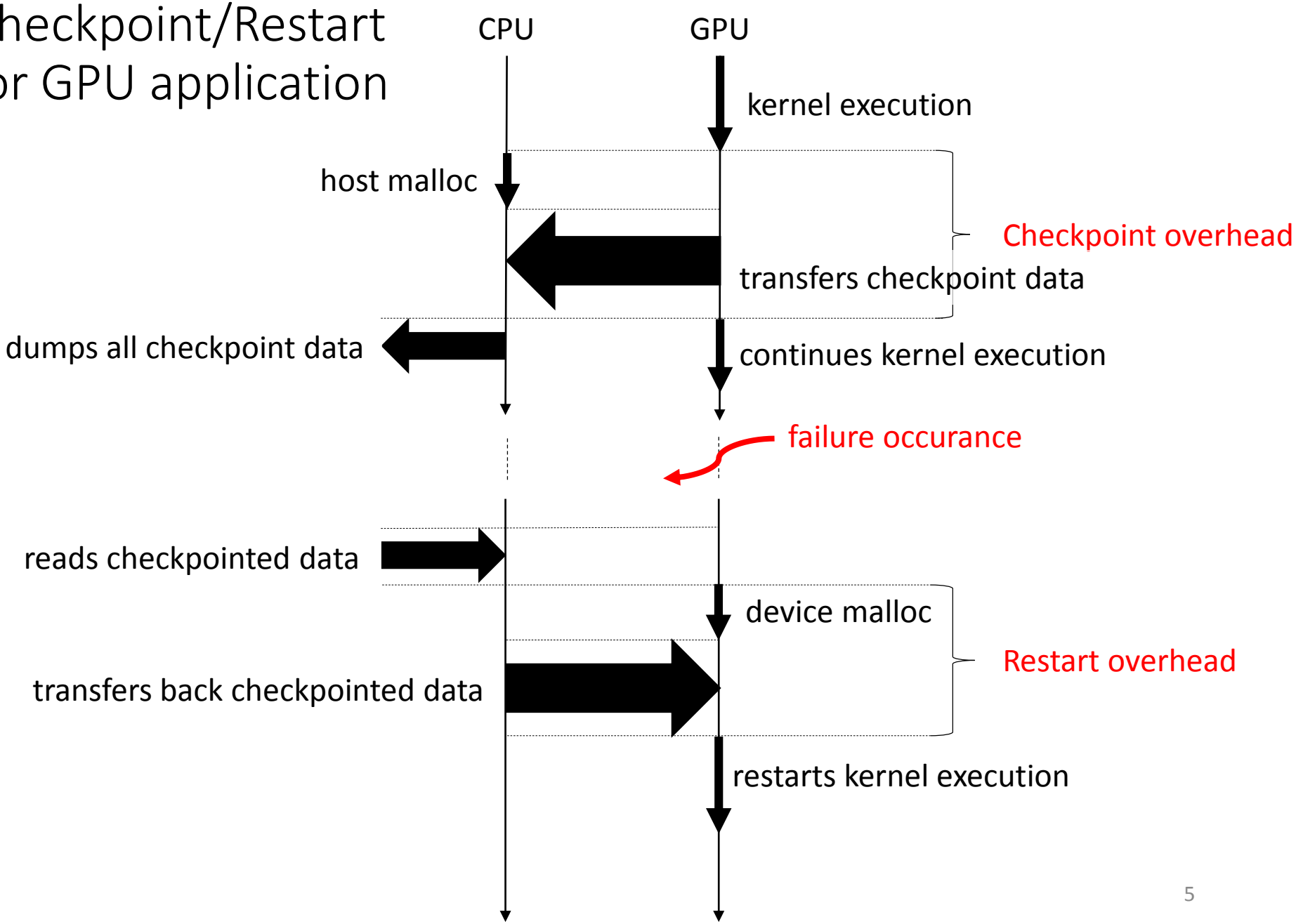


- **CUDA**
  - De facto standard for GPGPU
  - /usr/local/cuda on TSUBAME

Checkpoint/Restart for GPGPU

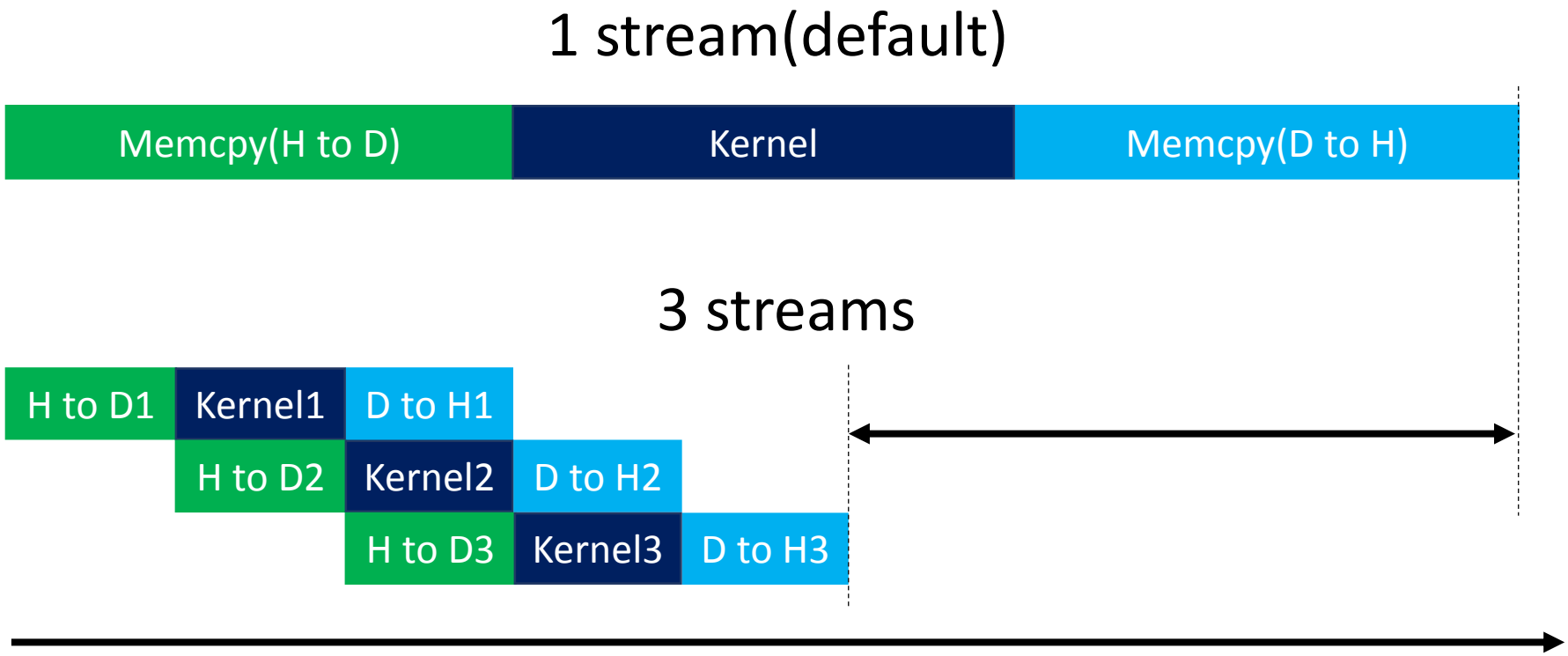
- BLCR[Paul, et al 2006]
  - Checkpoint/restart mechanism for Linux system
  - not works for GPU application
- VCCP[Hong, et al 2009]
  - Checkpoint/restart mechanism for virtual machines
  - may work
- NVCR[Nukada, et al 2011]
  - Transparent checkpoint/restart library for CUDA applications
  - not works for CUDA 4.0 ~
- CheCL[Takizawa, et al 2011]
  - Transparent checkpoint/restart library for OpenCL applications
  - API proxy may be able to apply to CUDA

Two-level Checkpoint/Restart for GPU application



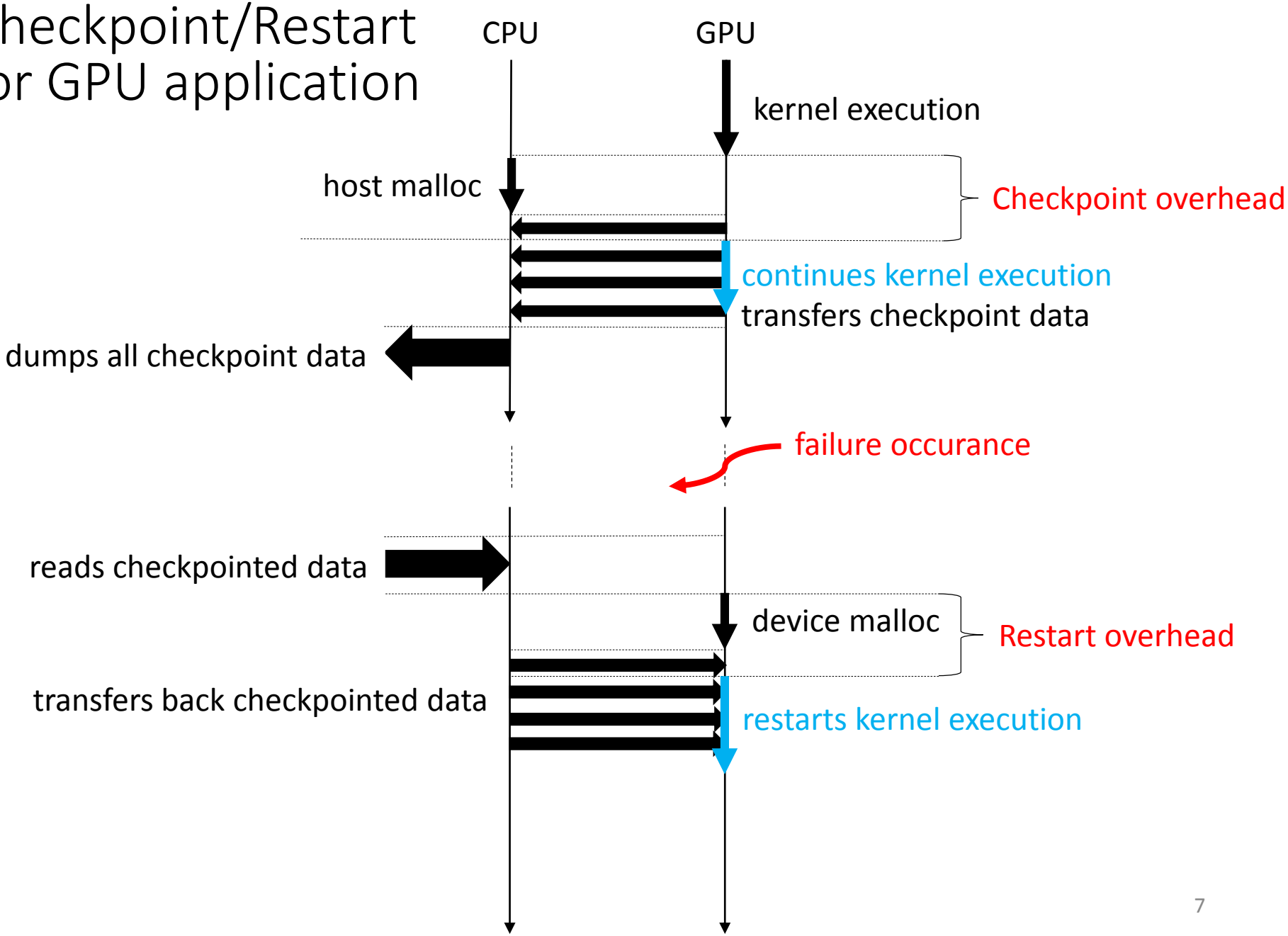
CUDA Stream

- For overlap communication and computation



/usr/local/cuda/samples/0_Simple/simpleStreams

Streamed Two-level Checkpoint/Restart for GPU application



Experiment

- NVIDIA GeForce GTX 295
  - compute capatibility 1.3
  - maximum number of blocks in a grid is 65535
  - maximum number of threads in block is 512
- Array addition C = A + B
  - Array size 2^{10} ~ 2^{24}
- Compare non-streamed, 4-streamed and 8-streamed

Experiment-1

… Checkpoint overhead

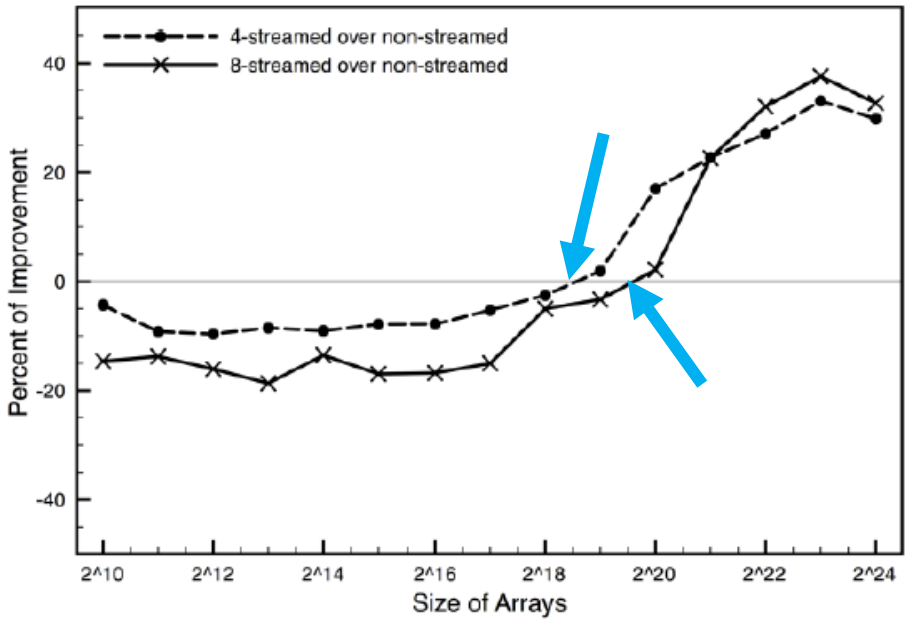
… Restart overhead

… Wasted time

- Pseudo code

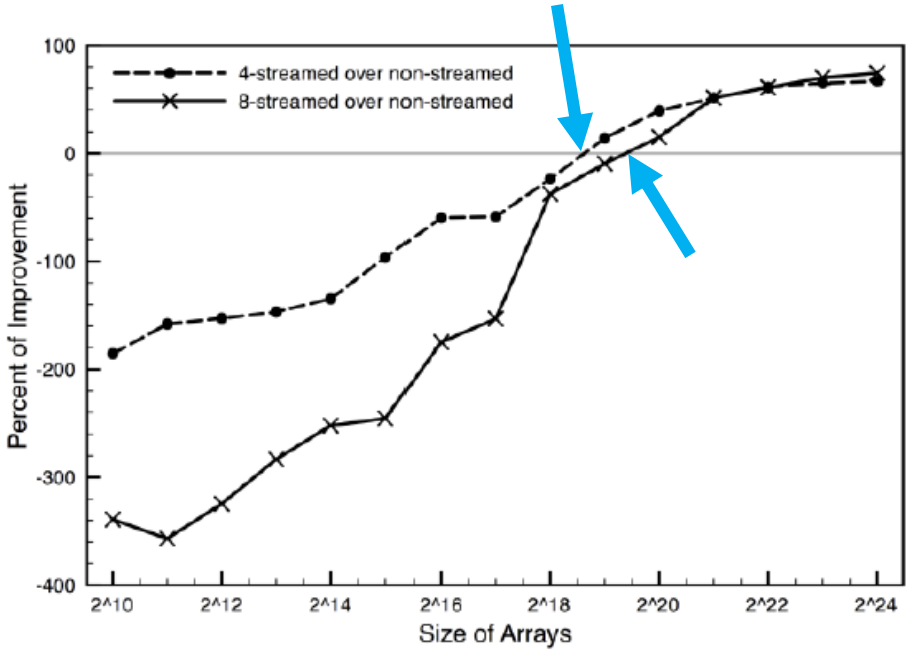
1. Execute kernel_1() with L iterations
2. Synchronize all threads to prepare for memory copy
3. Allocate host memory for data checkpointing, i.e. array A, B, and C.
4. Do device-to-host memory copy of array A, B, and C.
5. Execute kernel_2() with M iterations
6. Failure occurrence (Free all device memory)
7. Reallocate device memory for array A, B, and C.
8. Do host-to-device memory of array A, B, and C.
9. Re-execute kernel_2() with M iterations.
10. Execute kernel_3() with N iterations.
11. Do device-to-host memory copy of the result array C

Checkpoint overheads



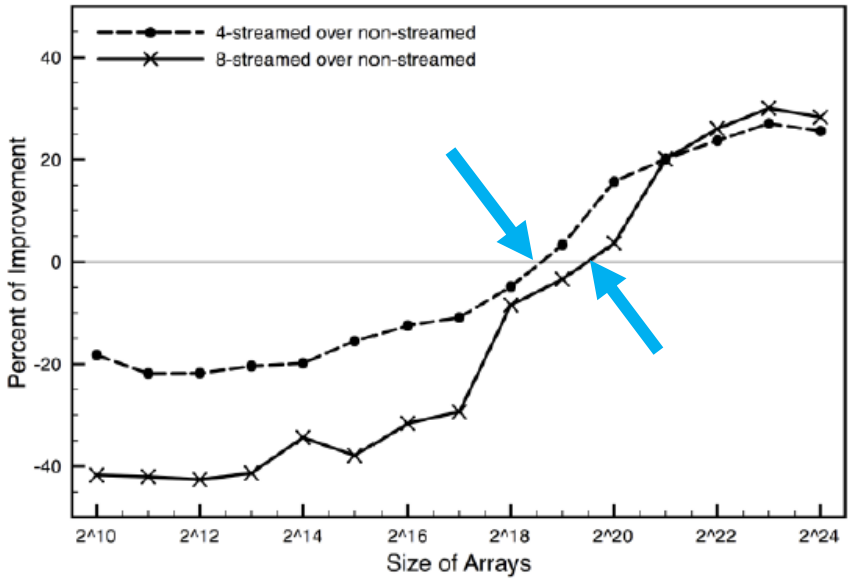
Array Size	Checkpoint Overhead (ms)		
	Non-streamed	4-streamed	8-streamed
2^{10}	5.116	5.337	5.812
2^{11}	5.086	5.555	5.795
2^{12}	5.098	5.590	5.924
2^{13}	5.159	5.600	6.108
2^{14}	5.239	5.713	5.940
2^{15}	5.404	5.829	6.285
2^{16}	5.679	6.123	6.640
2^{17}	6.256	6.586	7.196
2^{18}	7.390	7.577	7.770
2^{19}	9.579	9.396	9.917
2^{20}	14.475	12.016	14.171
2^{21}	22.726	17.568	17.613
2^{22}	41.075	29.938	27.930
2^{23}	80.754	54.034	50.436
2^{24}	147.704	103.702	99.220

Restart overheads



Array Size	Restart Overhead (ms)		
	Non-streamed	4-streamed	8-streamed
2^{10}	0.437	1.248	2.099
2^{11}	0.490	1.264	2.160
2^{12}	0.491	1.241	2.111
2^{13}	0.502	1.240	1.923
2^{14}	0.521	1.223	1.862
2^{15}	0.577	1.132	2.088
2^{16}	0.716	1.141	1.965
2^{17}	0.975	1.546	2.451
2^{18}	1.427	1.763	1.982
2^{19}	2.403	2.063	2.622
2^{20}	4.330	2.620	3.682
2^{21}	8.140	3.963	3.931
2^{22}	15.826	6.160	6.163
2^{23}	31.178	10.899	9.322
2^{24}	61.751	20.537	15.904

Wasted times



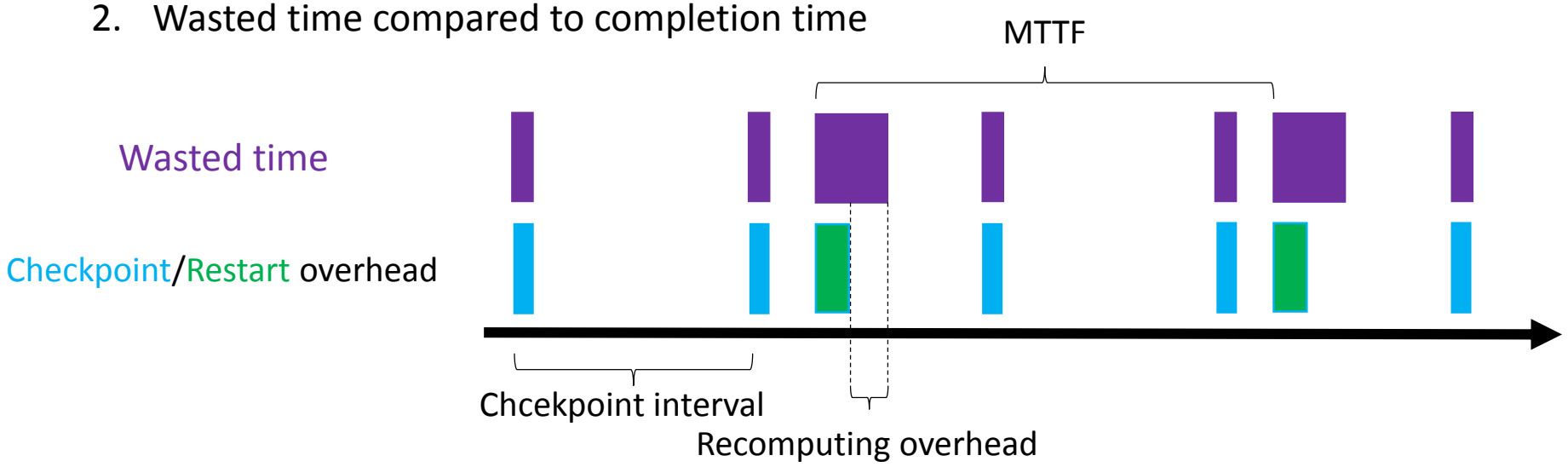
Array Size	Wasted Time (ms)		
	Non-streamed	4-streamed	8-streamed
2^{10}	5.665	6.698	8.025
2^{11}	5.688	6.931	8.067
2^{12}	5.701	6.943	8.147
2^{13}	5.776	6.954	8.146
2^{14}	5.936	7.112	7.978
2^{15}	6.322	7.302	8.714
2^{16}	6.964	7.834	9.175
2^{17}	8.255	9.156	10.671
2^{18}	10.803	11.326	11.738
2^{19}	15.913	15.389	16.469
2^{20}	26.573	22.404	25.620
2^{21}	46.352	37.016	37.029
2^{22}	87.749	66.946	64.941
2^{23}	173.624	126.625	121.450
2^{24}	332.888	247.673	238.557

Experiment-2

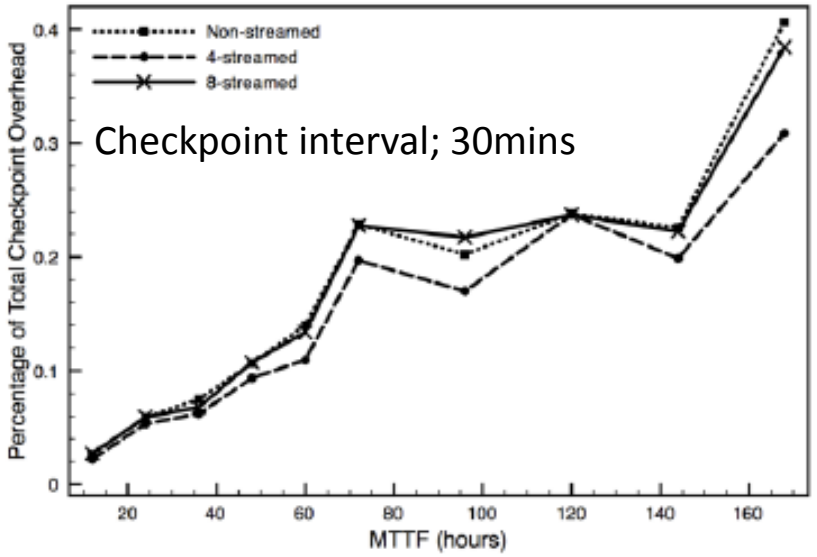
- Consider mean-time-to-failures(MTTFs)

Size of array	$2^{20}$and 2^{24}
MTTFs	12 hours to 7 days
Checkpoint interval	30 and 120 mins

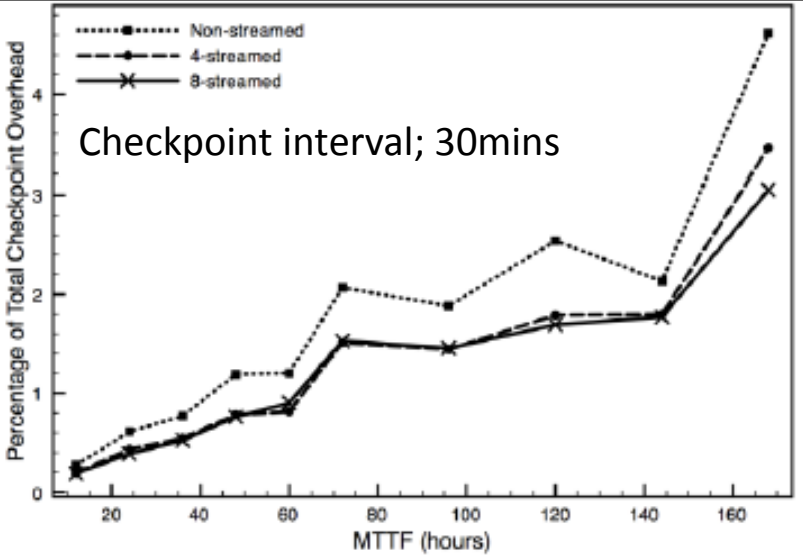
1. Checkpoint/restart overheads compared to wasted time
2. Wasted time compared to completion time



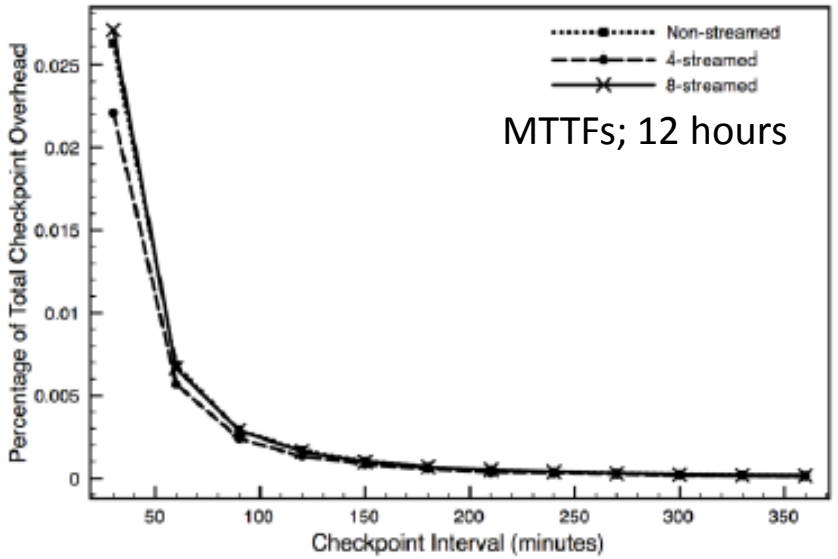
Checkpoint overhead/wasted time

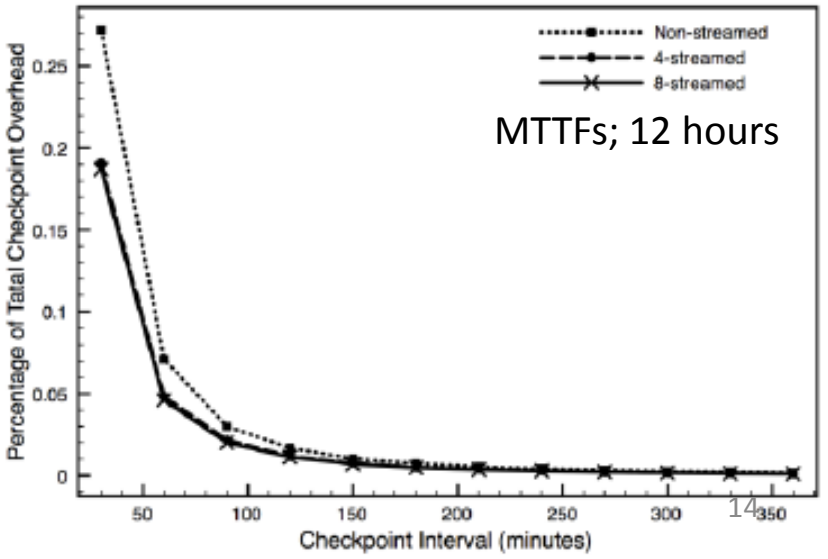


Array size 2^{20}

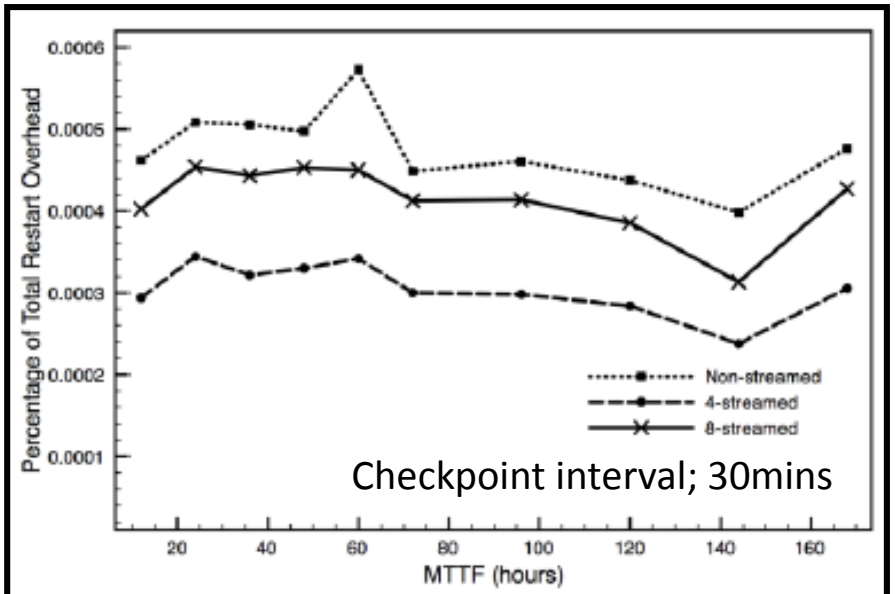


Array size 2^{24}

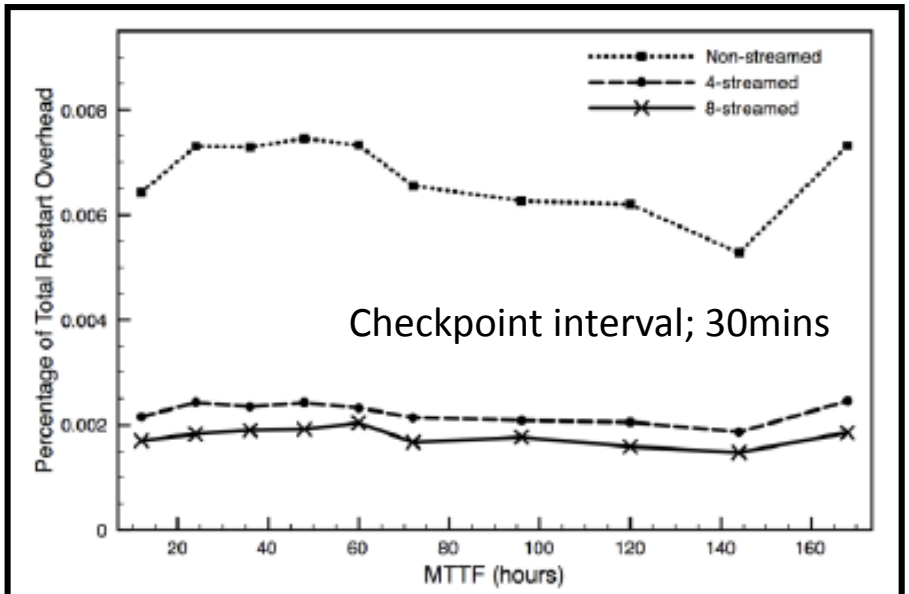




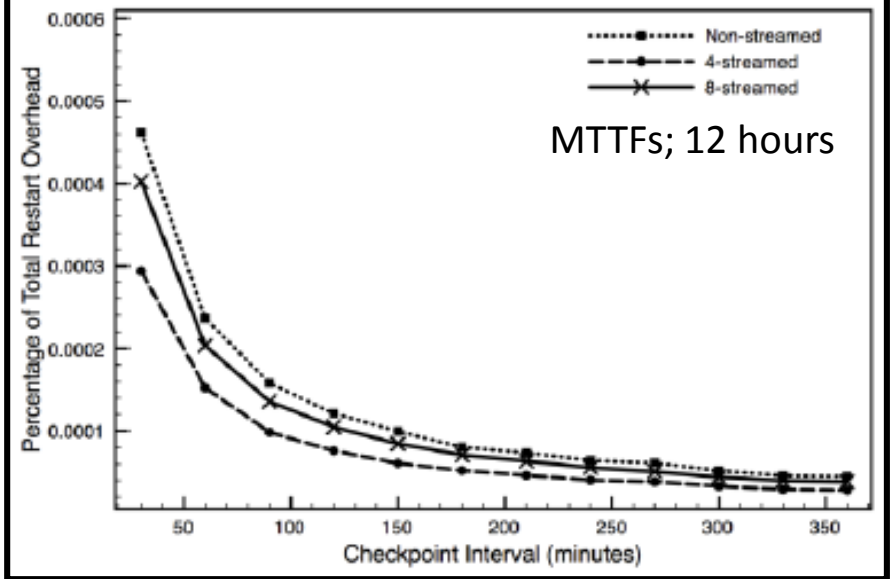
Restart overhead/wasted time

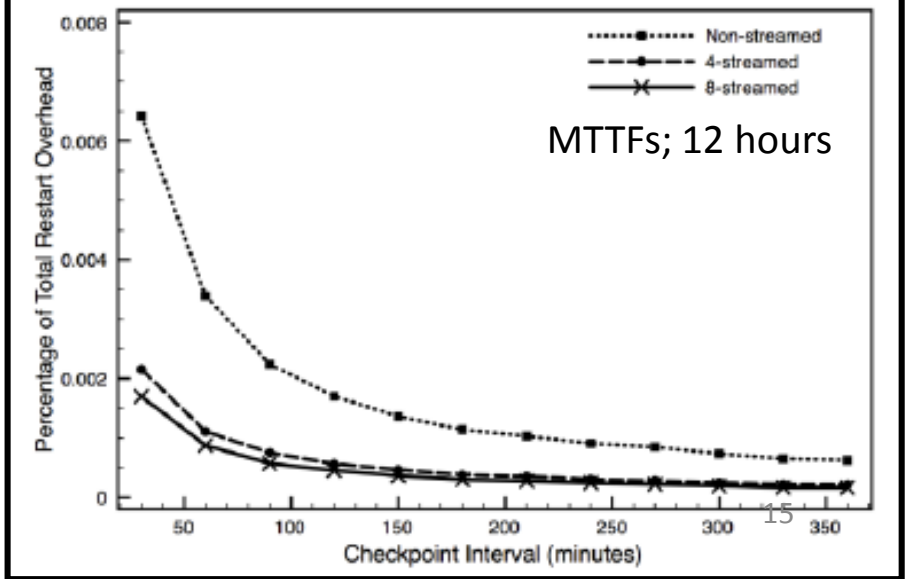


Array size 2^{20}

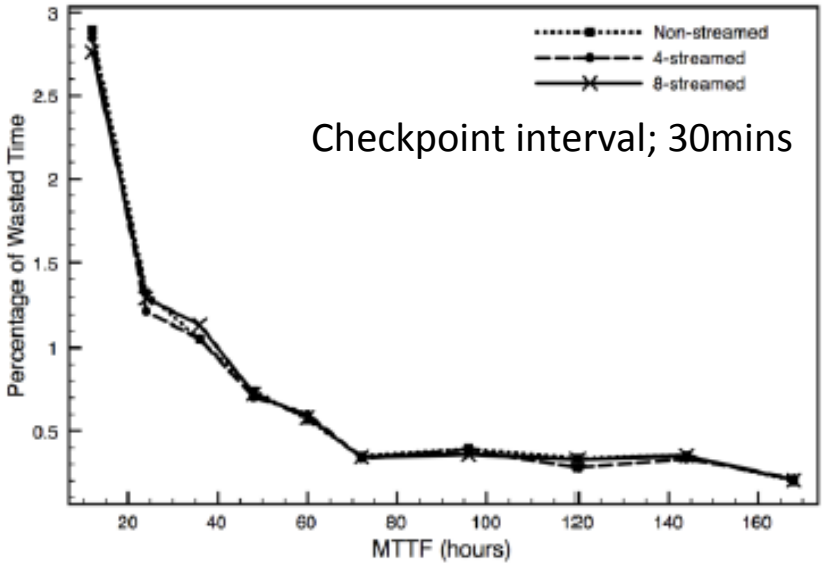


Array size 2^{24}

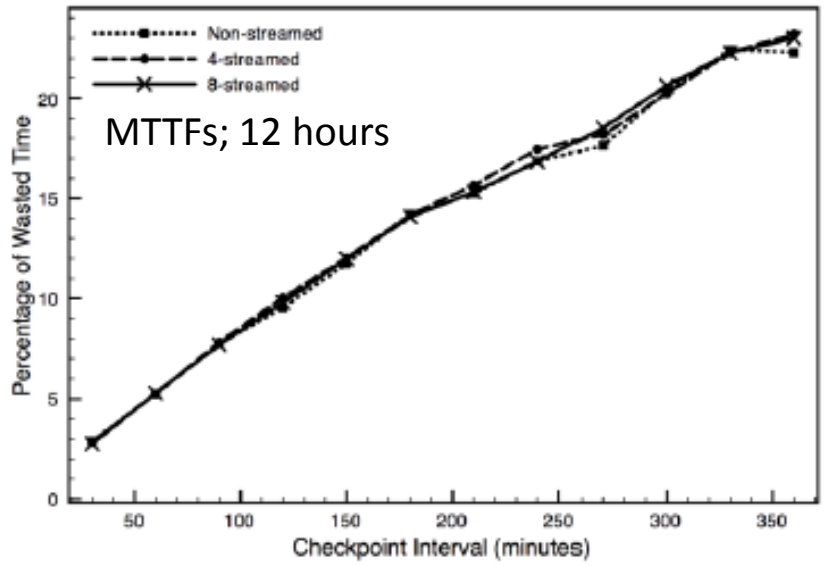


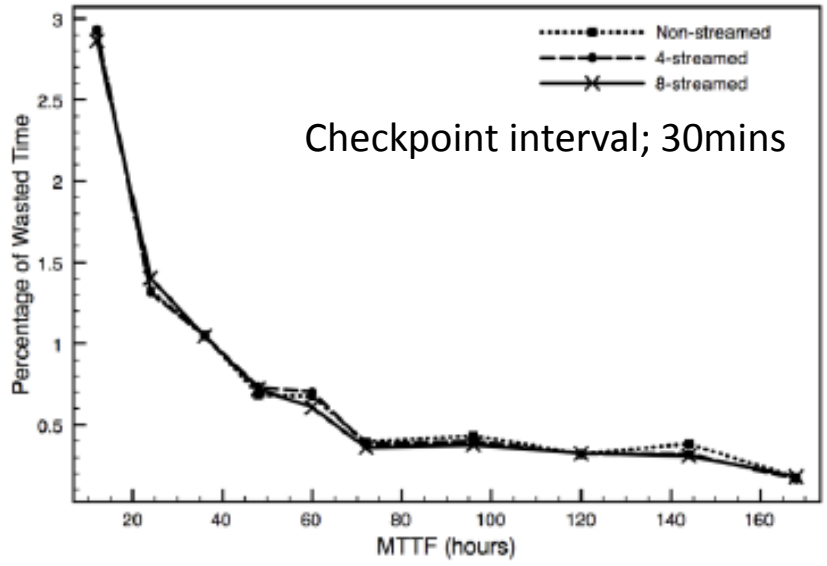


Wasted time/completion time

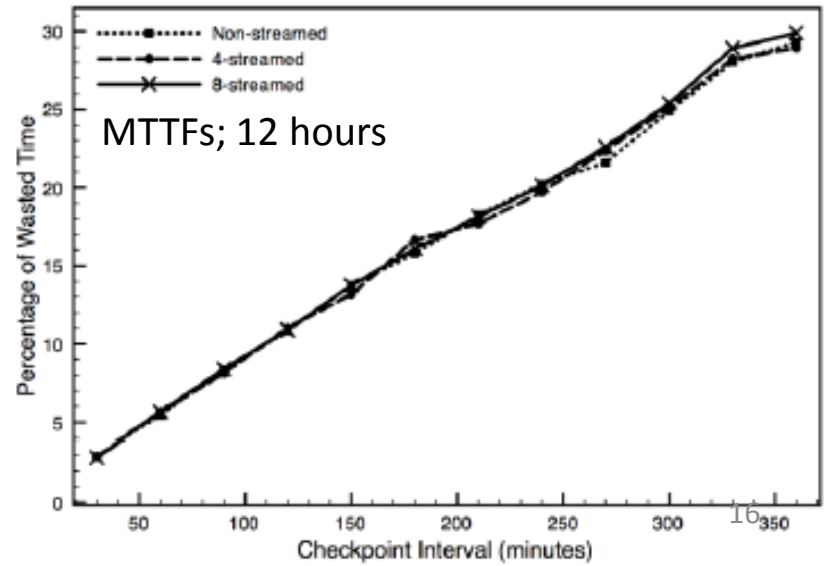


Array size 2^{20}





Array size 2^{24}



Conclusion

- GPGPU
  - GPUs for non-graphic applications
  - De facto standard; CUDA
- Checkpoint/restart modeling for GPGPU
  - Two-level
  - Streamed
- Streamed Checkpoint/Restart model has advantage over non-streamed model when data size is enough big

Discussion

- How to implement the streamed model?
  - especially for an application with overlapped kernel?

- The results of experiments are affected by a characteristic of application.