GPU Accelerated Gravitational Wave Discovery

Zhihui Du
Department of Computer Science and Technology,
Tsinghua University, Beijing China
Joint work with
The University of Western Australia
And
Academy of Mathematics and System Science, CAS

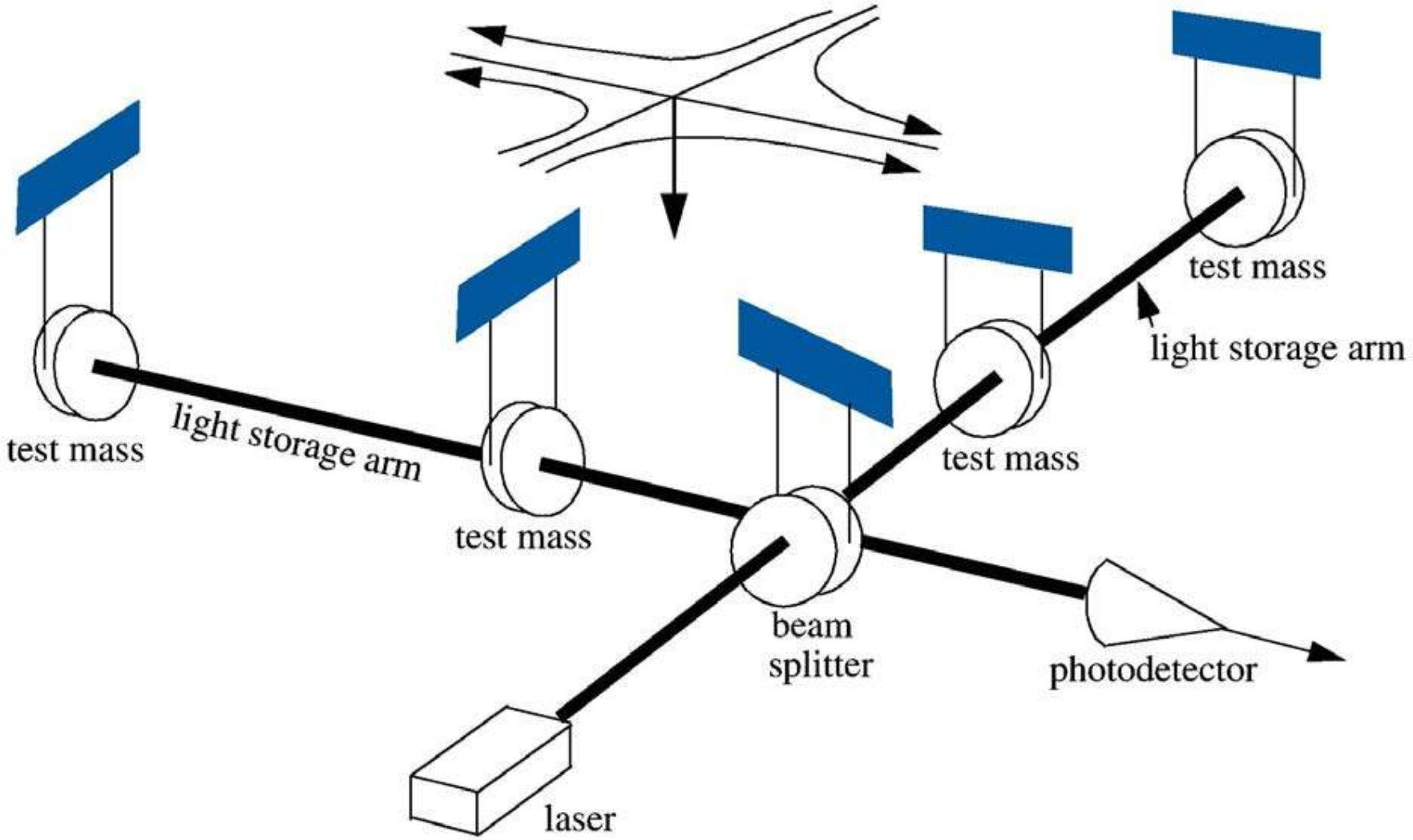
Outline

- Background
- Challenges
- Achievements
- Methods and Experimental Results
- Conclusion & future work

Gravitational Waves

- Predicted to exist by Albert Einstein in 1916 on the basis of general relativity (but have not been detected till now)
  - open a completely new window to the universe
    - "listen" to violent events from early universe such as The Big Bang.
    - detect the formation of black holes
- Nobel Prize
  - 1993, Russell Hulse and Joseph Taylor (*indirect* evidence for its existence)
  - New Nobel Prize (*directly* detected)
- GW Observatories
  - LIGO (America), Virgo (France and Italy), GEO 600 (Germany and Great Britain) and TAMA 300-DECIGO (Japan)

Interferometer Detectors for Gravitational Waves




LIGO


LIGO


Virgo


GEO600


TAMA

Two Challenging Problems

- How to model GW sources accurately?
  - Know what GWs look like exactly
  - The simulation (only method to model the GWs) is time-consuming
    - 7 months
- How to process GW data quickly?
  - Detect GW quickly to allow prompt electromagnetic follow-up observations
  - Processing the data as fast as possible
    - delay less than 1 s

Achievements

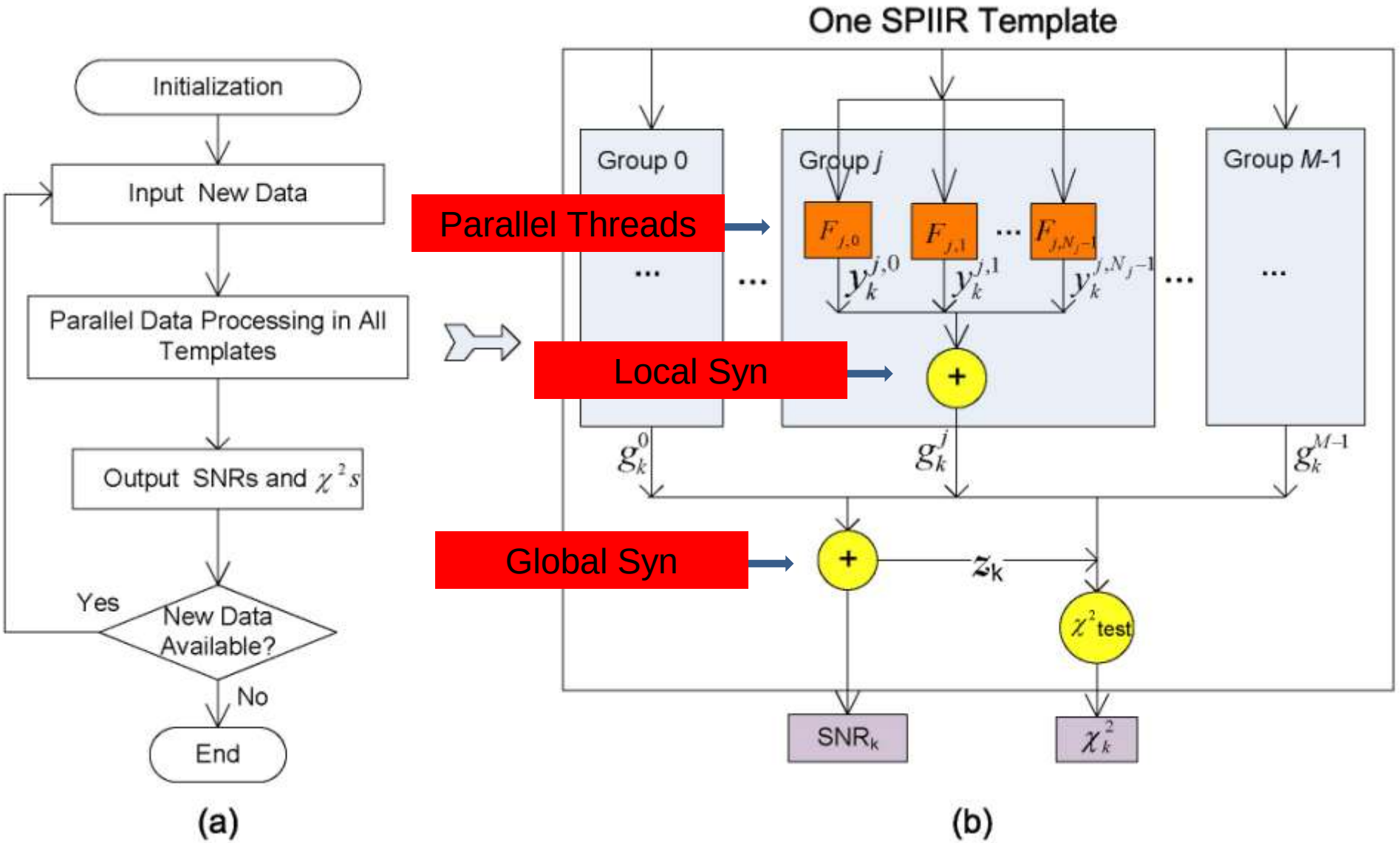
- GW source modeling (5X speedup)
  - Compared with parallel MPI program on the same supercomputer Tianhe-1A (straightforward implementation)
  - 7 months -> 1.4 month
- GW data processing (50X speedup)
  - Compared with sequential CPU program on desktop computer
  - Real time data processing on a PC

Our Methods

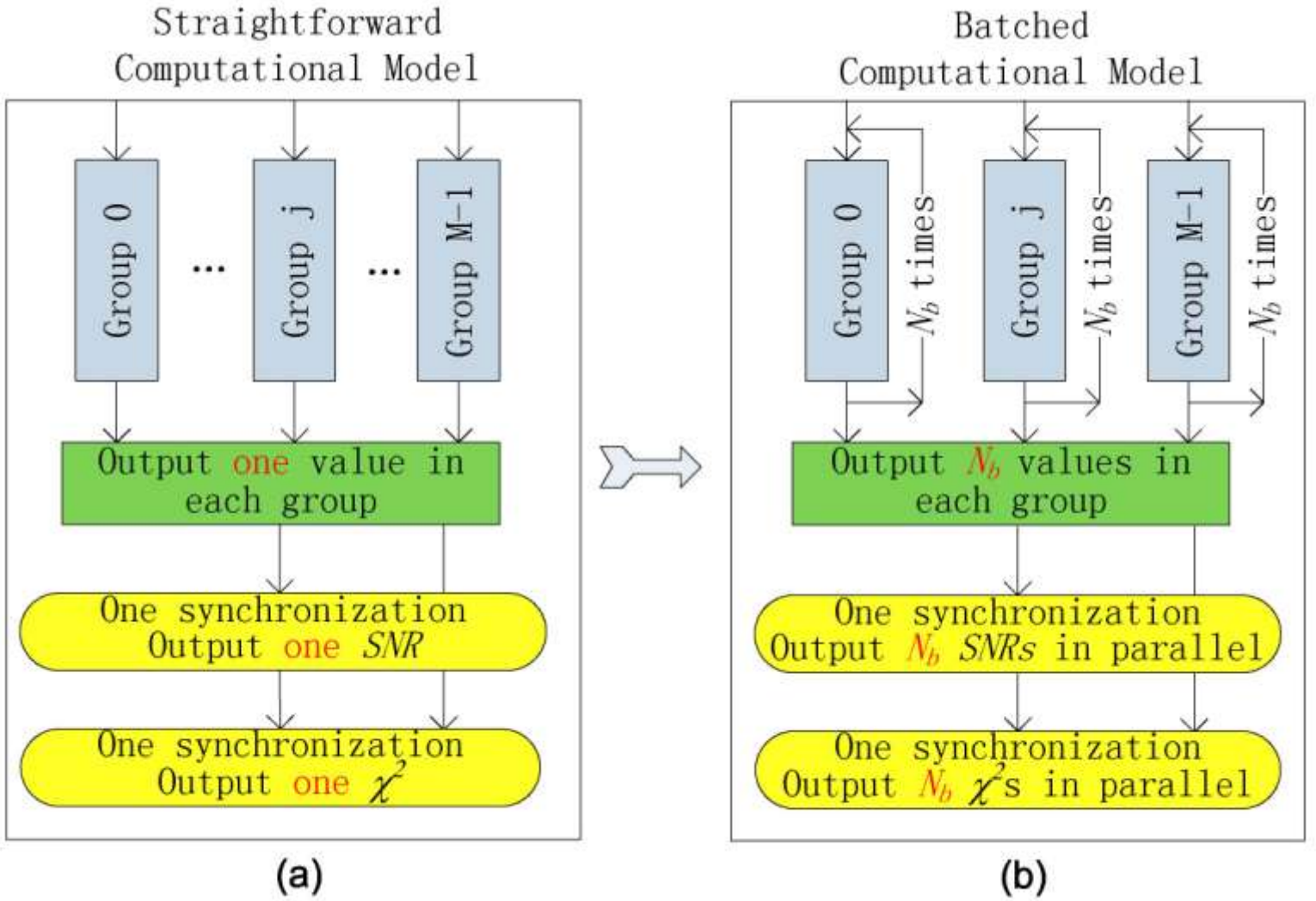
- Identify the computational intensive parts
  - Runge-Kutta method (RK4)
  - Summed Parallel Infinite Impulse Response filtering(SPIIR)
- Rewrite those parts in CUDA on GPU

SPIIR Method Processing Loop

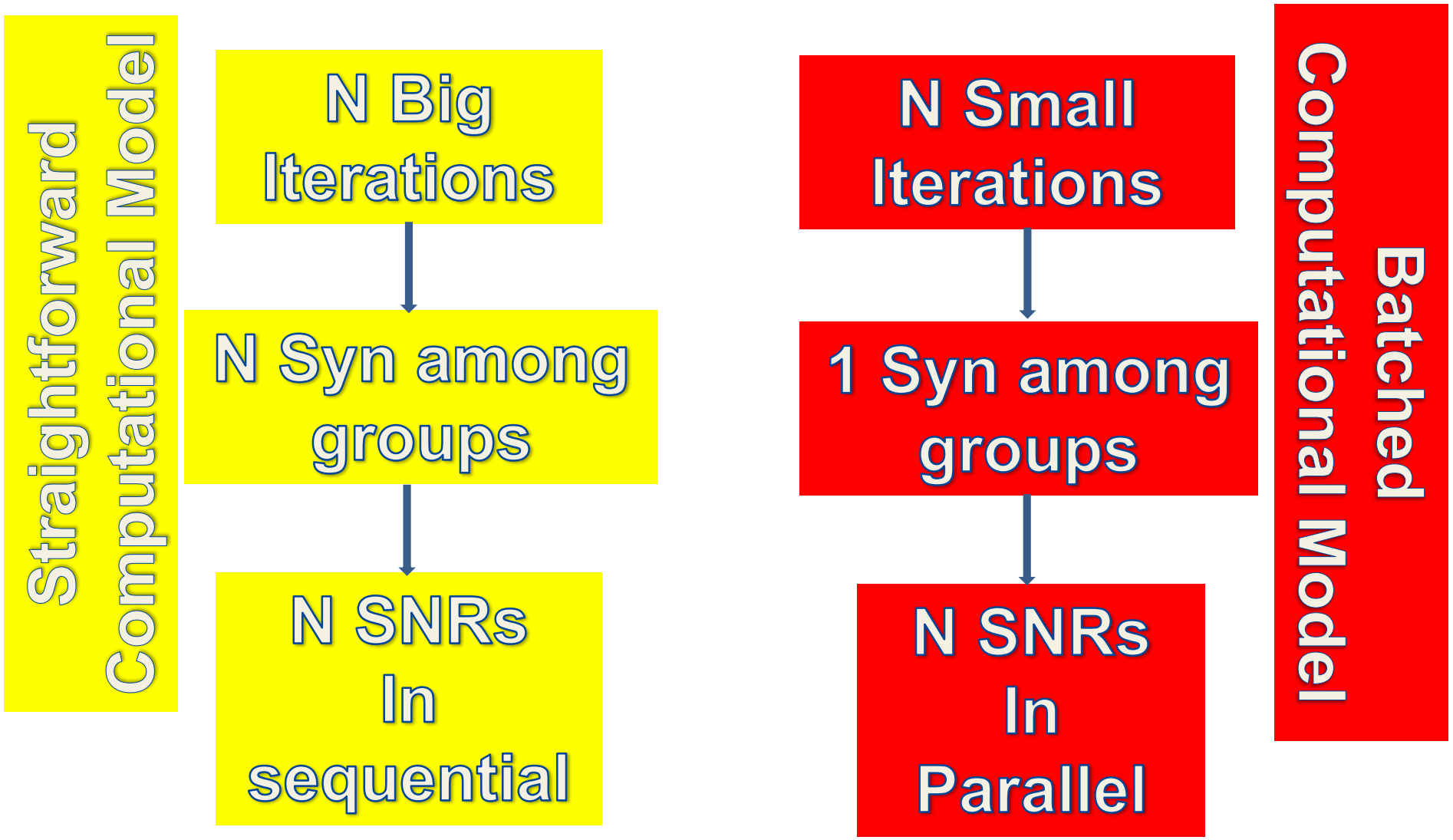
(Summed Parallel Infinite Impulse Response)



Batched Computational Model to Reduce #(Global Syn)

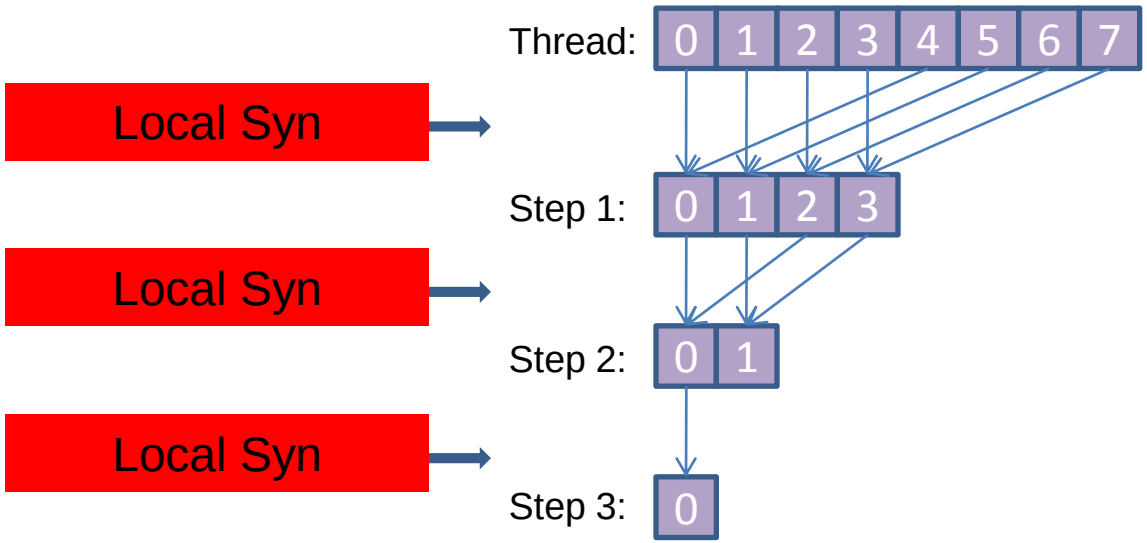


Comparison

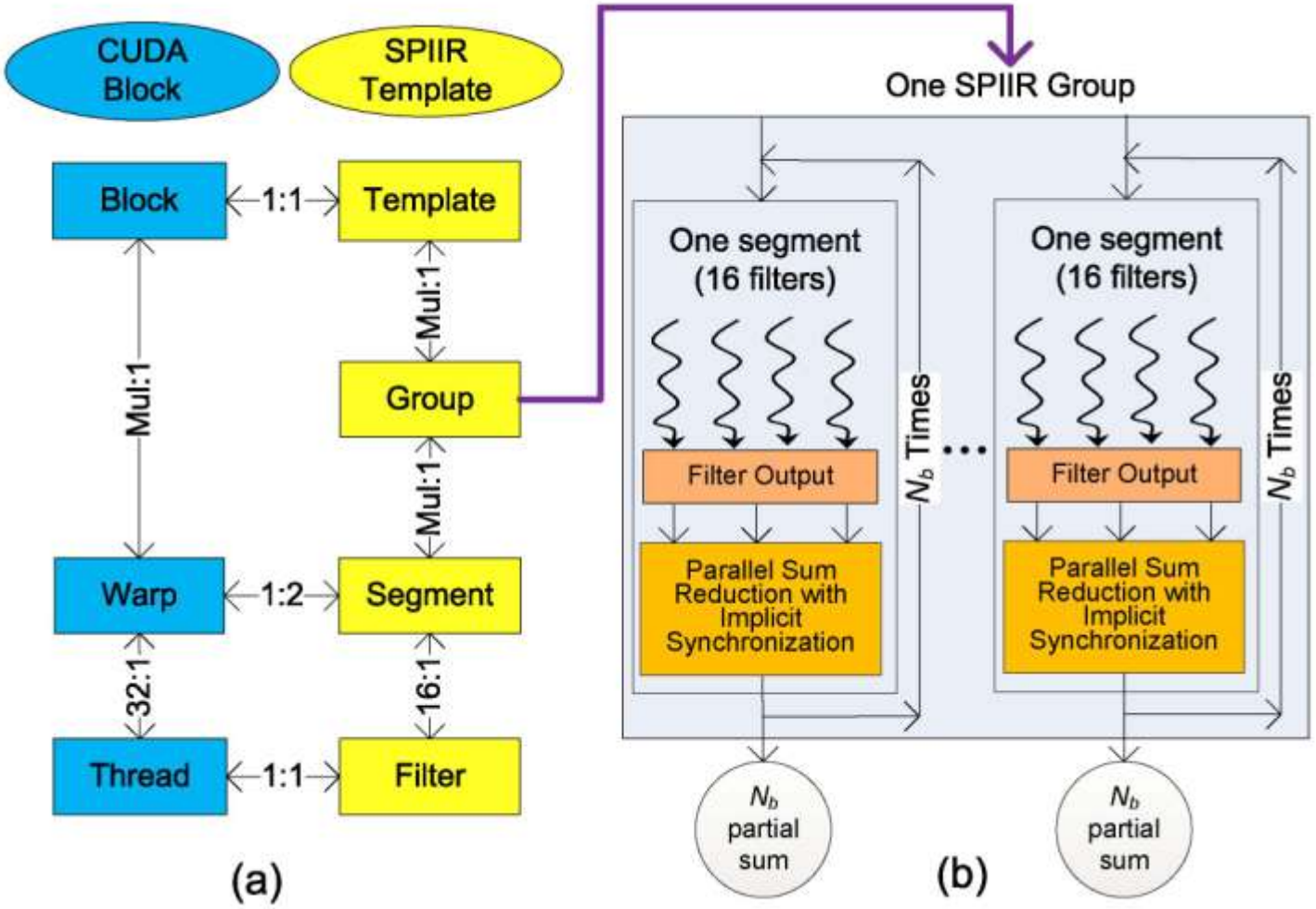

Straightforward Computational Model
N Big Iterations
N Syn among groups
N SNRs In sequential
Batched Computational Model
N Small Iterations
1 Syn among groups
N SNRs In Parallel

Summation Methods

- Sequential Methods
  - N number, one thread, N-1 add steps
- Parallel sum reduction (improve parallelism,2x speedup)
  - N number, N threads, $\log_2 N$ add steps

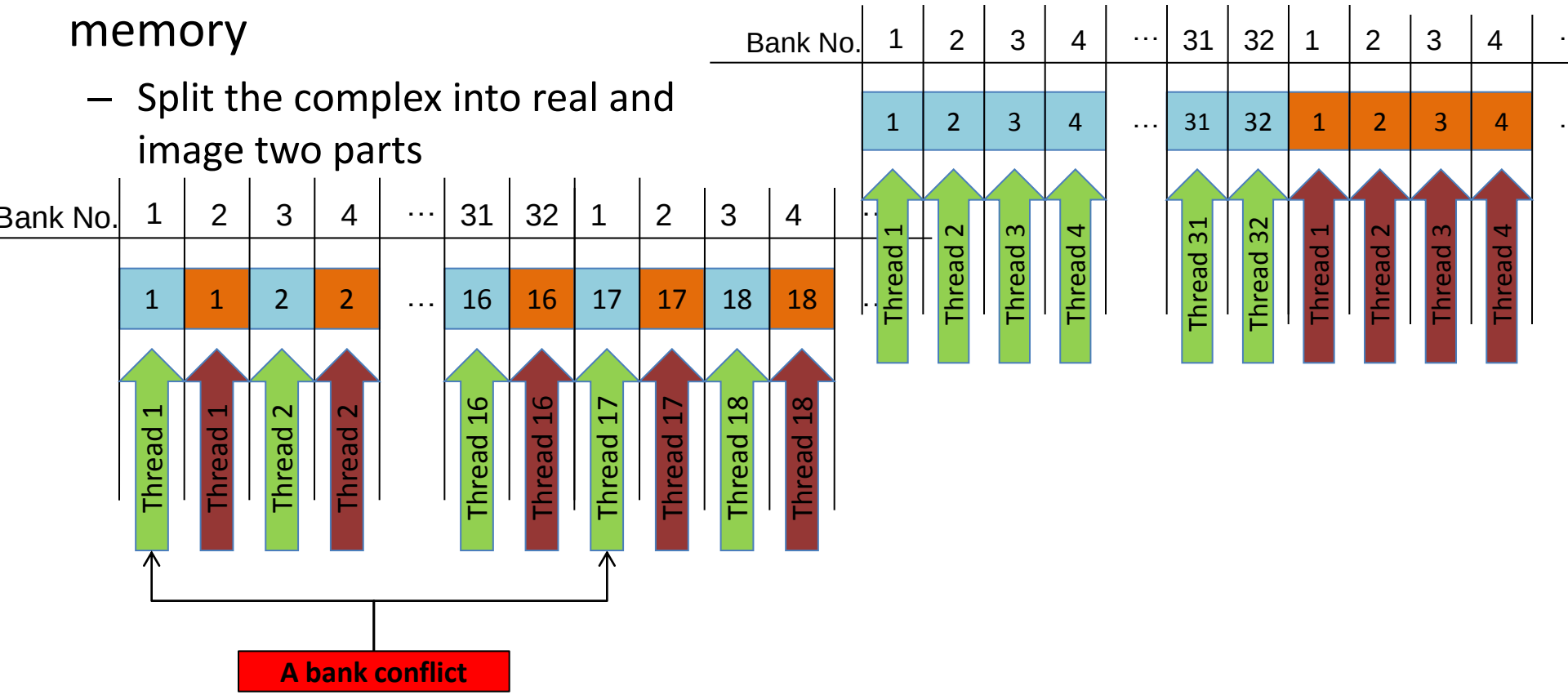


Remove Local Synchronization in Parallel Sum Reduction



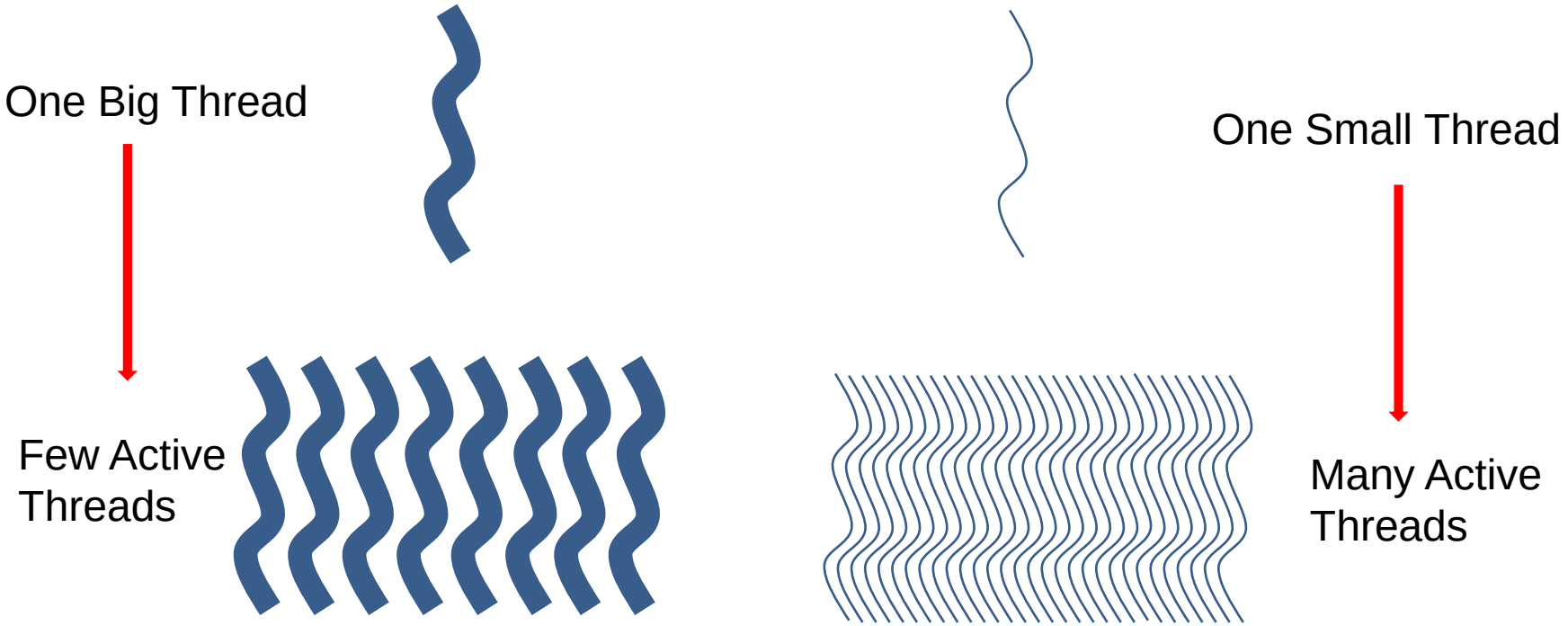
Memory Optimization

- Texture Cache
  - Take advantage of high access performance of texture cache and the locality of our application
  - 1.43x performance gain
- Rearrange the data structure to avoid bank conflicts in shared memory
  - Split the complex into real and image two parts



Improve the hardware utilization

- Adjust register and shared memory usage to improve occupancy (active threads)



Experimental Results

Hardware	CPU	Intel *Core* i7 920 2.67 GHz
	GPU	NVIDIA *GeForce* GTX 480
	Host memory	6 GB DDR3
Software	Operating system	Fedora 16 64-bit
	CUDA version	4.1
	Host compiler	gcc 4.6.3

Method	Overall speedup
Straightforward	7.0
Batched+PSR-IS	25
+Texture memory	42
+Avoid bank conflicts	48
+Tuning resource usage	58

Application features based methods

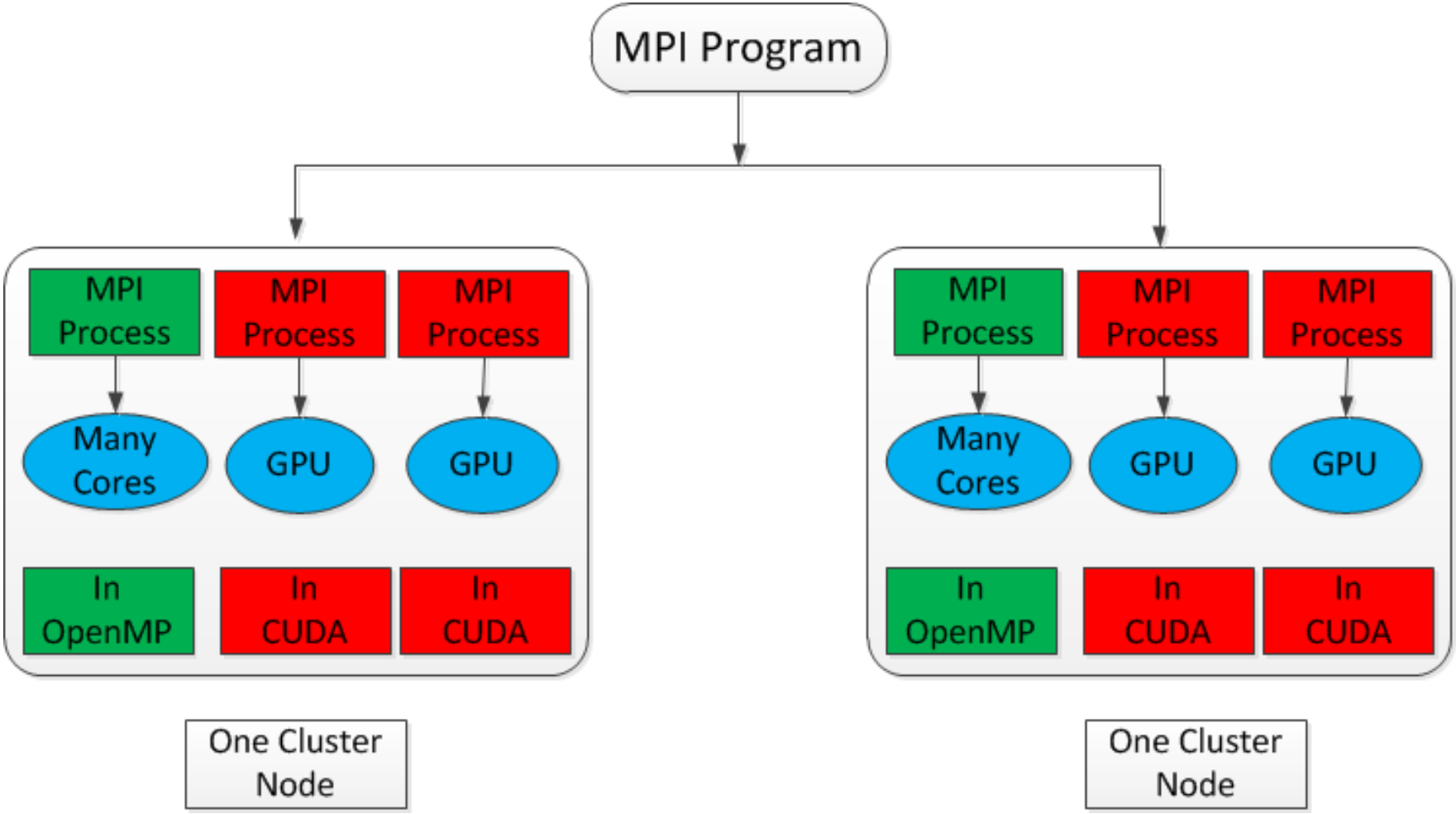
General methods

Half of the speedup from the hardware based optimization, another half of the speedup from the application based optimization

Numerical precision

- GPU_FLOAT gave roughly 0.002% average fractional errors on the SNR values and 0.0002% on the χ^2 values
- 99% of the SNR and the χ^2 values had less than 0.007% and 0.001% errors respectively.

Accelerate AMSS-NCKU on GPU Cluster



Conclusion & Future work

- GPU can really help to improve the performance of computational intensive part
- MPI+OpenMP+CUDA can explore the full performance of GPU clusters

Thanks

Q&A