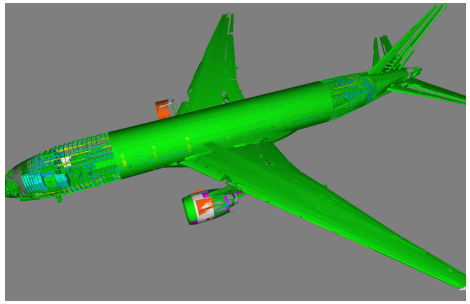
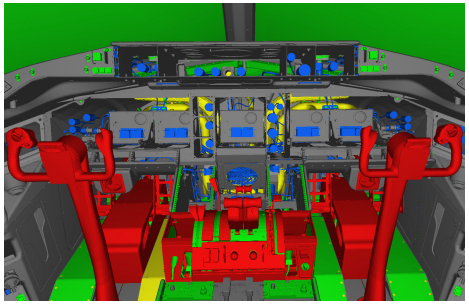
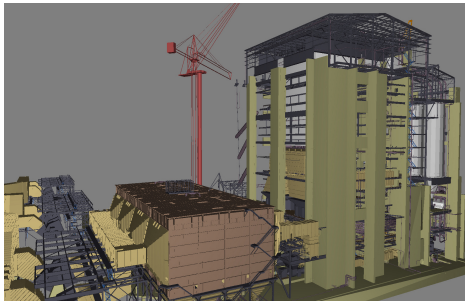
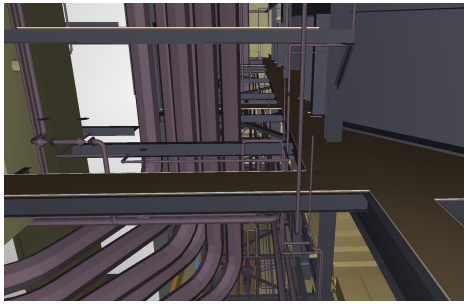
GPU TECHNOLOGY CONFERENCE 2013 | MARCH 18-21, 2013 | SAN JOSE, CALIFORNIA

GPU TECHNOLOGY CONFERENCE

Multi-GPU Load Balancing for Simulation and Rendering

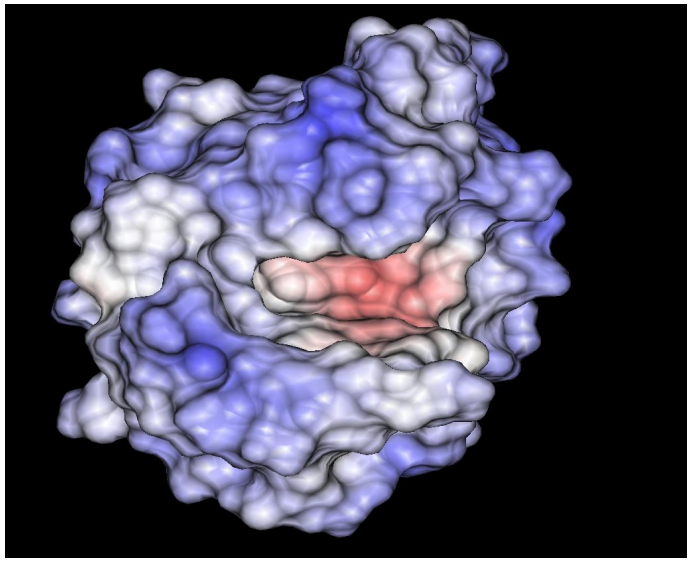
Yong Cao

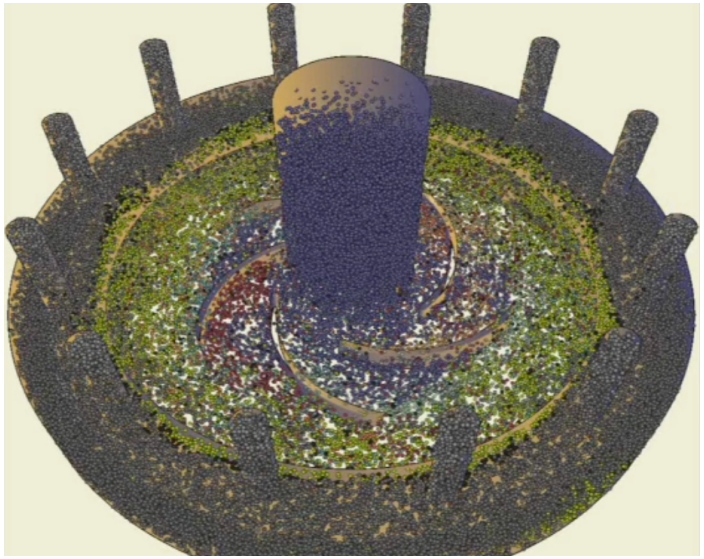
Computer Science Department, Virginia Tech, USA

In-situ Visualization and Visual Analytics

- Instant visualization and interaction of computing tasks
- Applications:
  - Computational Fluid Dynamics
  - Seismic Propagation
  - Molecular Dynamics
  - Network Security Analysis
  - …





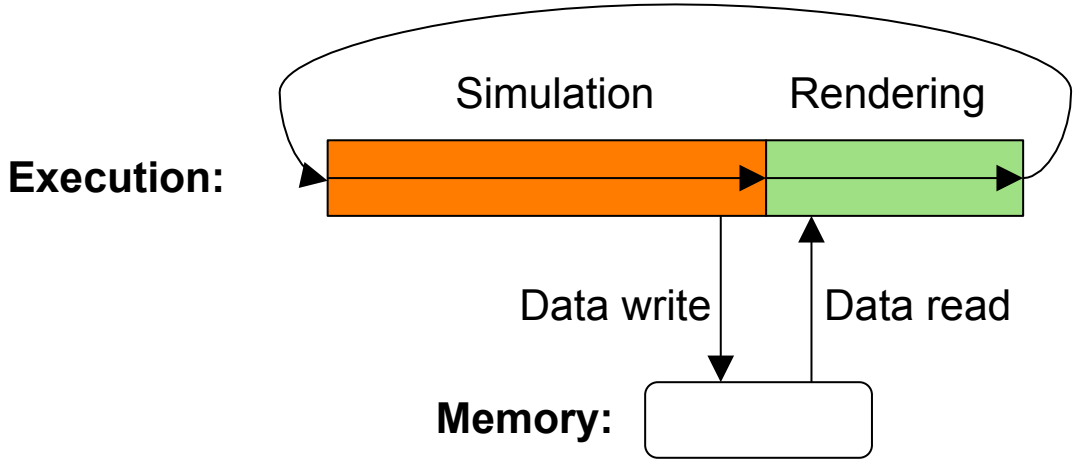
In-situ Visualization and Visual Analytics

- Instant visualization and interaction of computing tasks
- Applications:

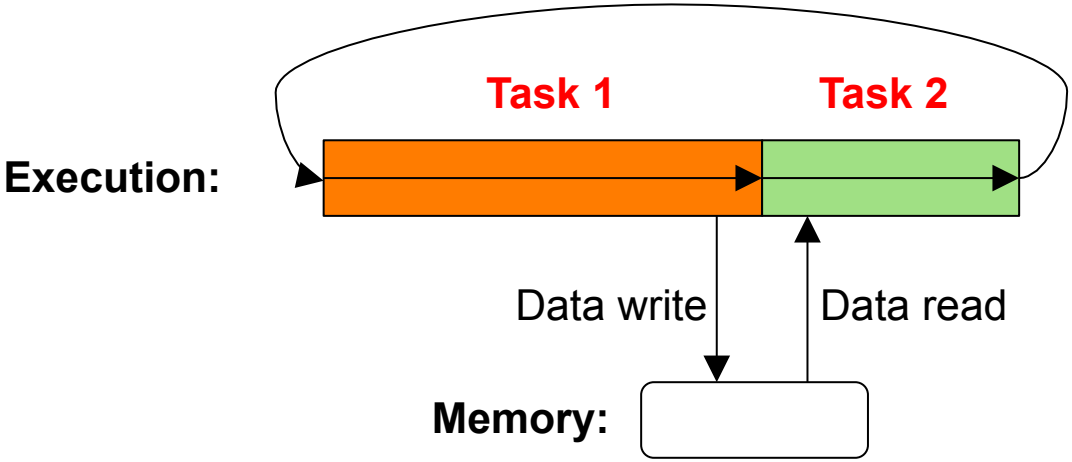
In-situ Visualization and Visual Analytics

- Instant visualization and interaction of computing tasks
- Applications:

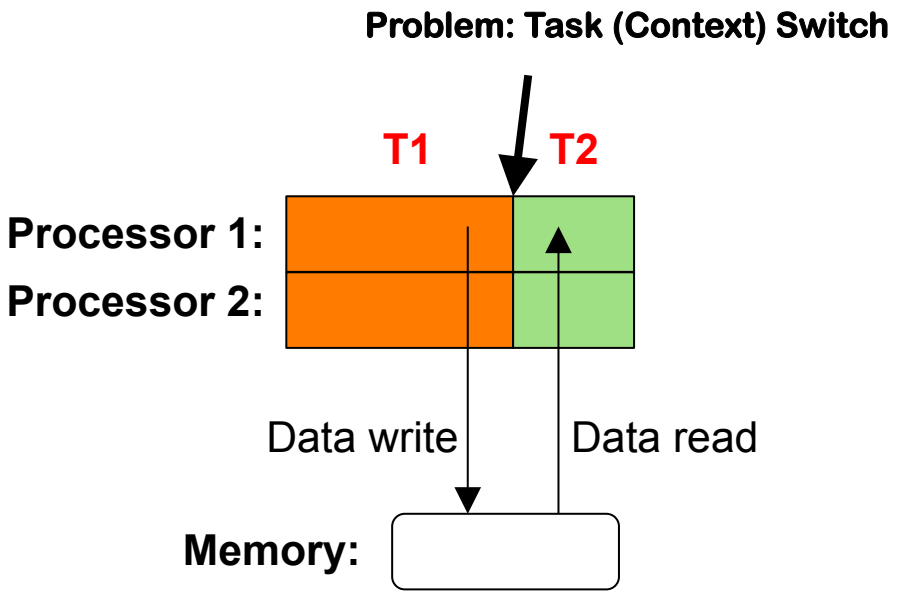
Generalized Execution Loop



Generalized Execution Loop

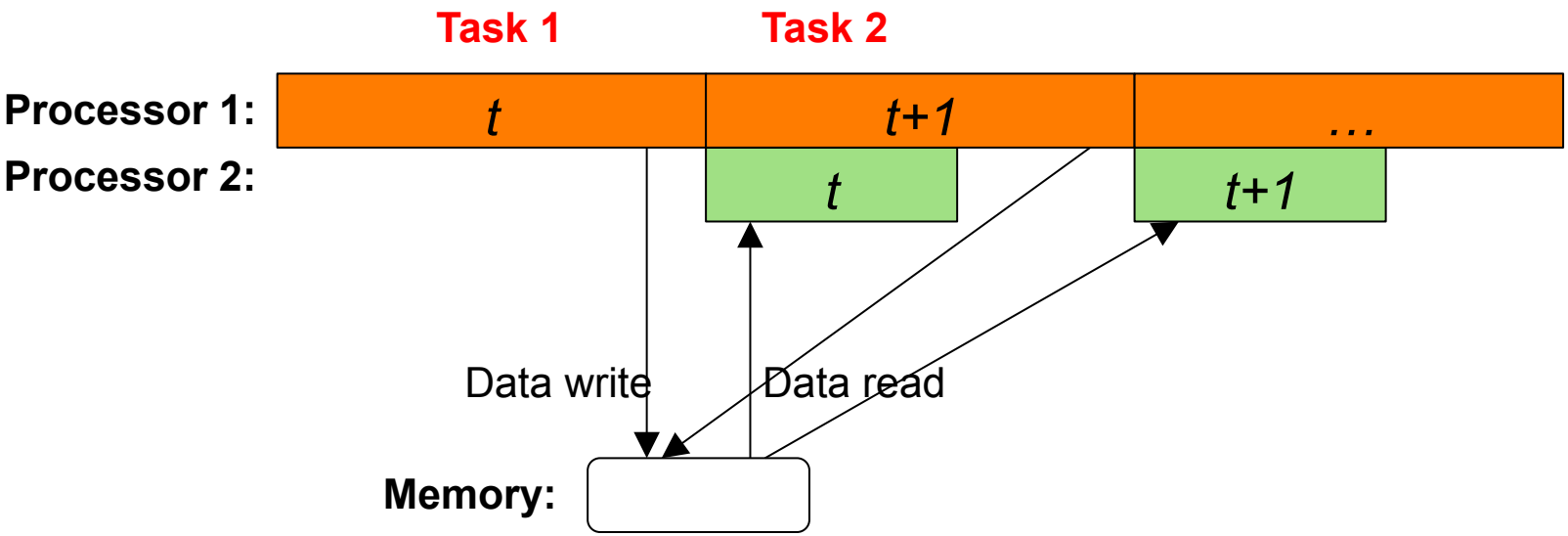


Parallel Execution – Task Split



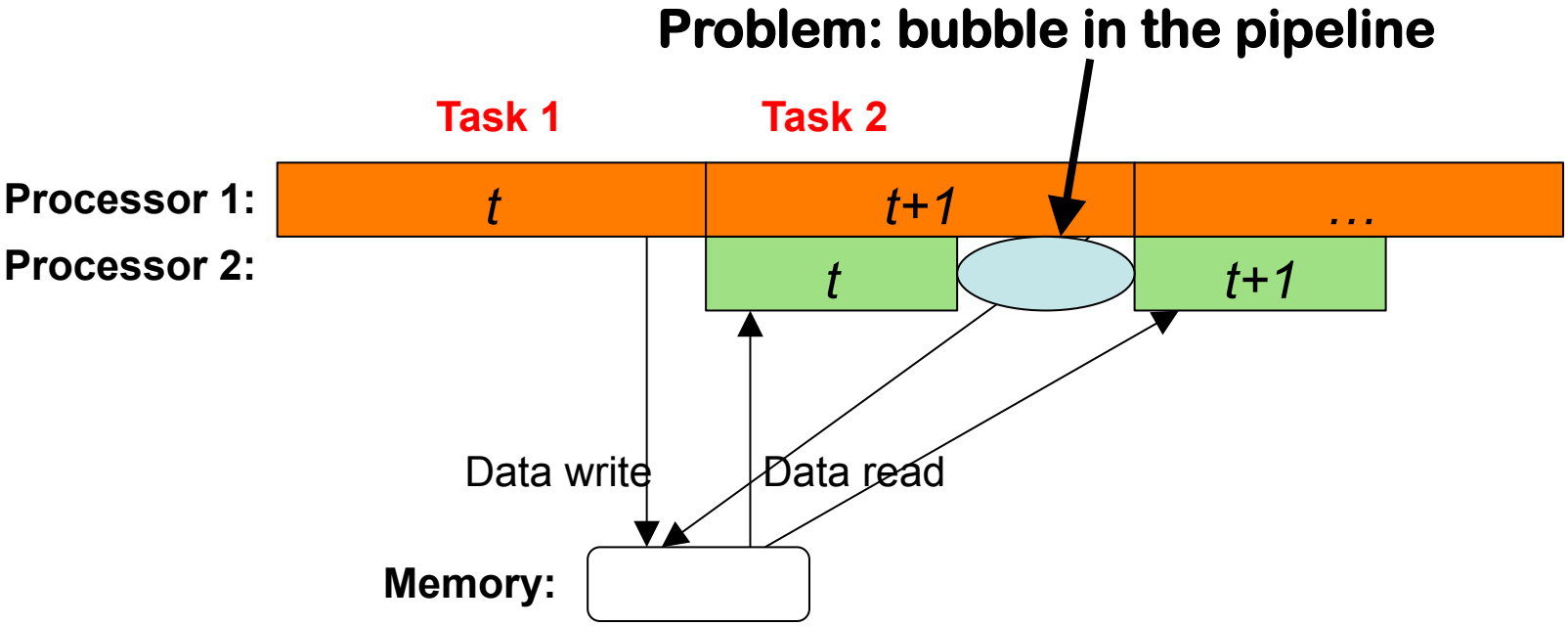
- Disadvantage of context switch:
  - Overhead of another kernel launch
  - Flash of the cache lines
  - Disallow persistent threads

Parallel Execution: Pipelining

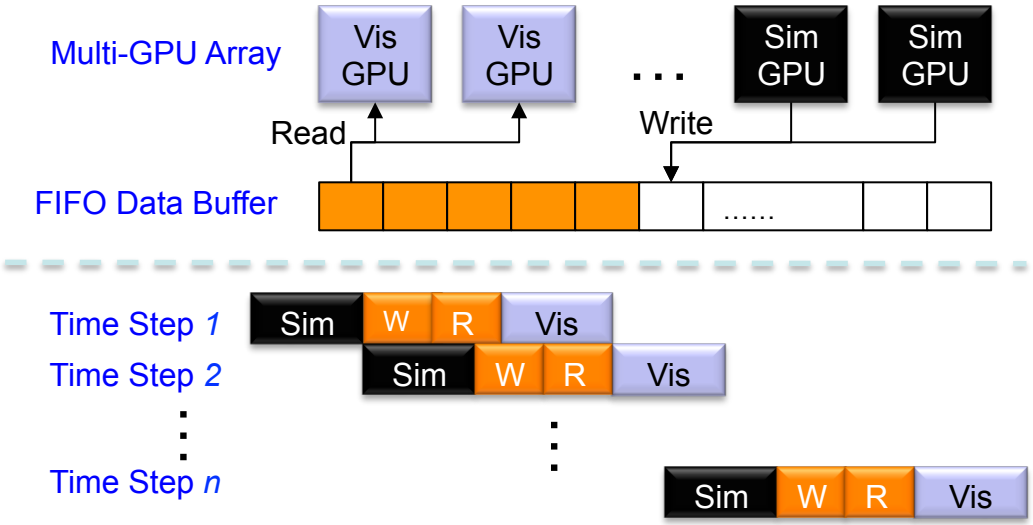


+ Simplified kernel for each GPU
+ Better share memory and cache usage
+ Persistent thread for distributed scheduling

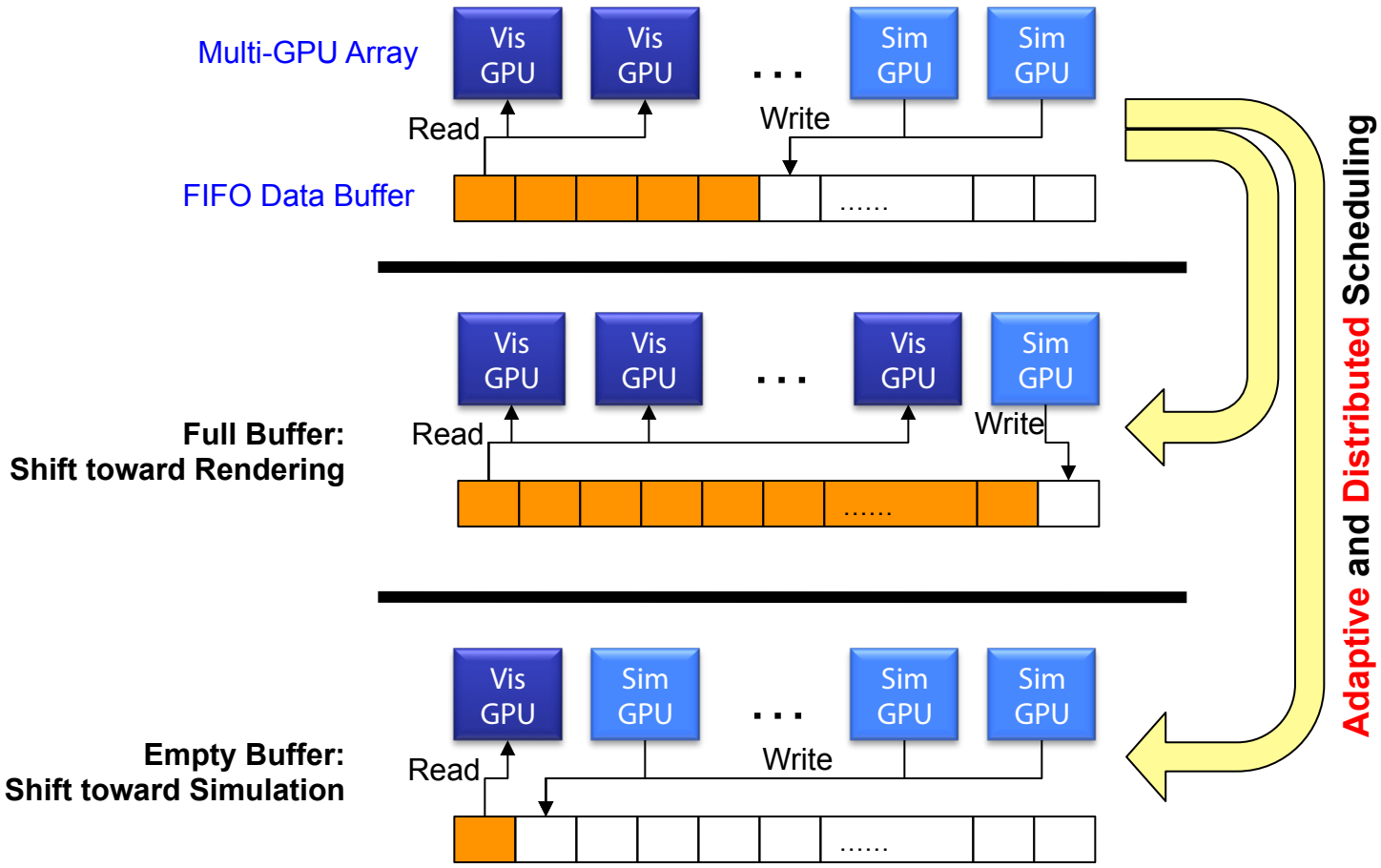
Parallel Execution: Pipelining



Multi-GPU Pipeline Architecture

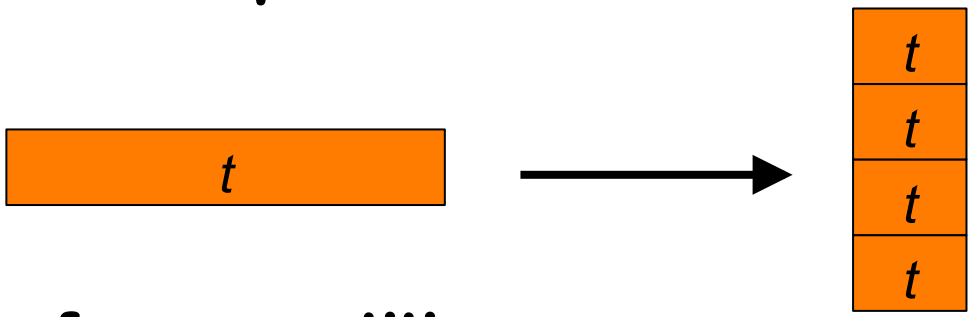


Adaptive Load Balancing

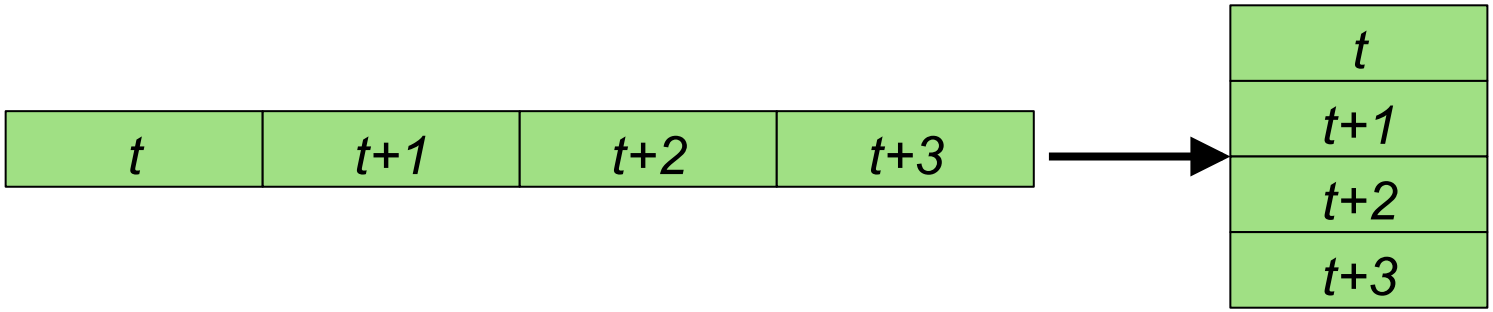


Task Partition

- **Intra-frame partition**

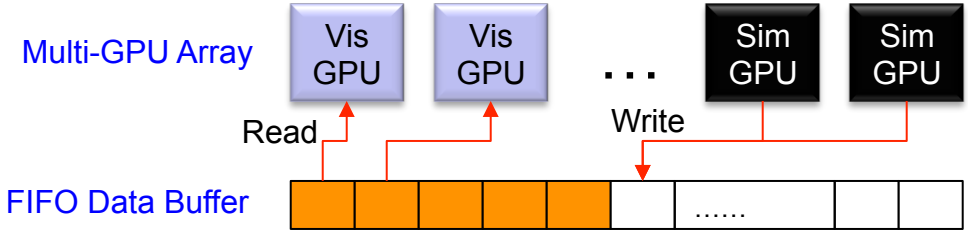


- **Inter-frame partition**



Task Partition for Visual Simulation

- **Simulation: Intra frame partition**
- **Rendering: Inter frame partition**



Problem: Scheduling Algorithm

- **Performance Model:**

$$T_i^{task} = \psi_i^{task}(n, p_1, p_2, \dots)$$

n: The number of assigned GPUs.

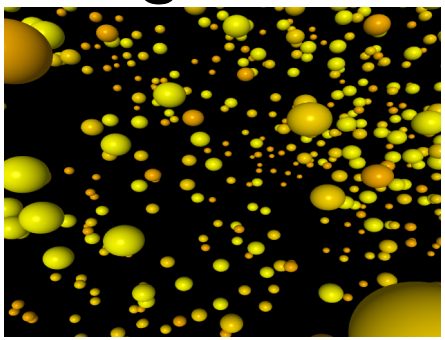
- **Schedule to optimize:**

$$M_i = \underset{M_i \in \{1,\dots,N-1\}}{argmin} (\|\psi_i^{sim}(M_i, p_1, p_2, \dots) - \psi_i^{vis}(N - M_i, p_1', p_2', \dots)\|)$$

M_i: The number of assigned Simulation GPUs.

Case Study Application

- **N-body Simulation with Ray-Traced rendering**



- **Performance model parameters:**

$$T_i^{task} = \psi_i^{task}(n, p_1, p_2, \ldots)$$

  - Simulation: number of iterations (i)
    number of simulated bodies (p)
  - Rendering: number of samples for super sampling (s)

- **Scheduling Optimization:**

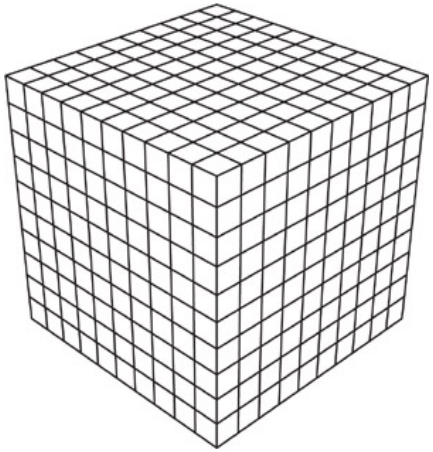
$$M_t = f(i_t, s_t, p_t)$$

Static Load-Balancing

- **Assumption: the performance parameters do NOT change at run-time.**

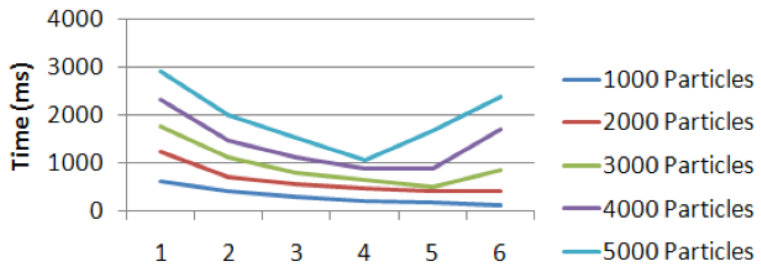
$$M_t = f(i_t, s_t, p_t) \longrightarrow M = f(i, s, p)$$

- **Data driven modeling approach:**
  - Sample the 3 dimensional (i,s,p) as a rigid grid
  - Use tri-linear interpolation to get the result for the new inputs

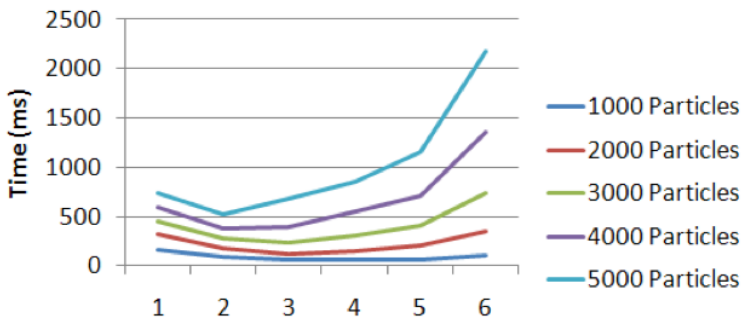


Static Load-Balancing: Results

- **Performance Parameter Sampling**

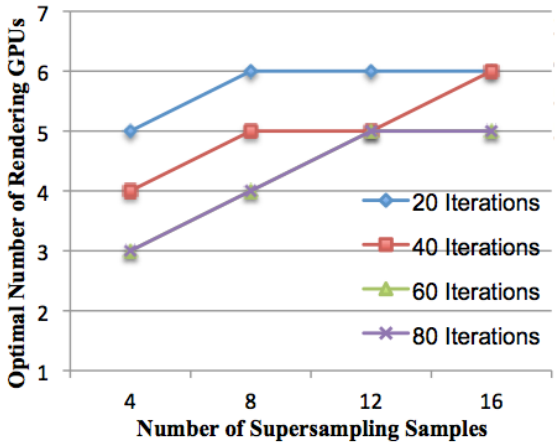


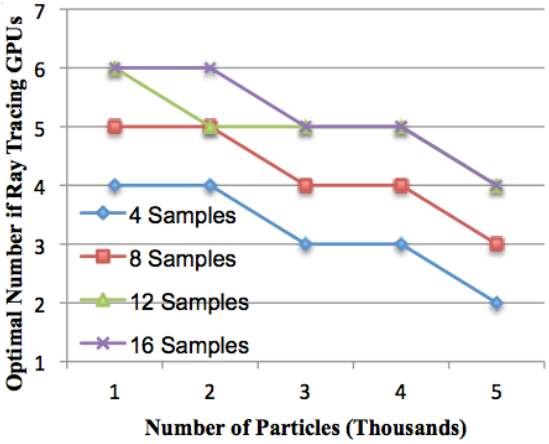
16 Samples, 80 iterations



4 Samples, 80 iterations

- **Load Balancing**





Dynamic Load Balancing

- **Assumption: Performance parameters change during the run-time.**

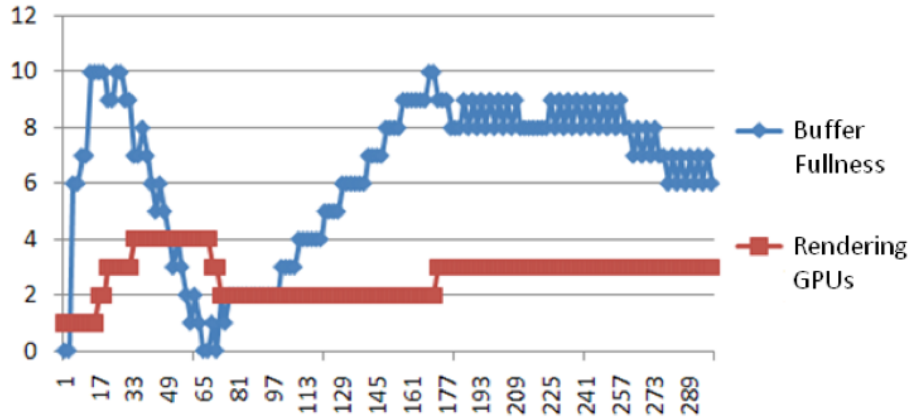
$$\psi_i^{sim}(M_i, p_1, p_2, \dots) - \psi_i^{vis}(N - M_i, p_1', p_2', \dots) \approx \phi(p)$$

- **Find the indirect load-balance indicator p**
  - Execution time of the previous time step
    - Problem: Performance different between two time steps can be dramatic.
  - The fullness of the buffer F

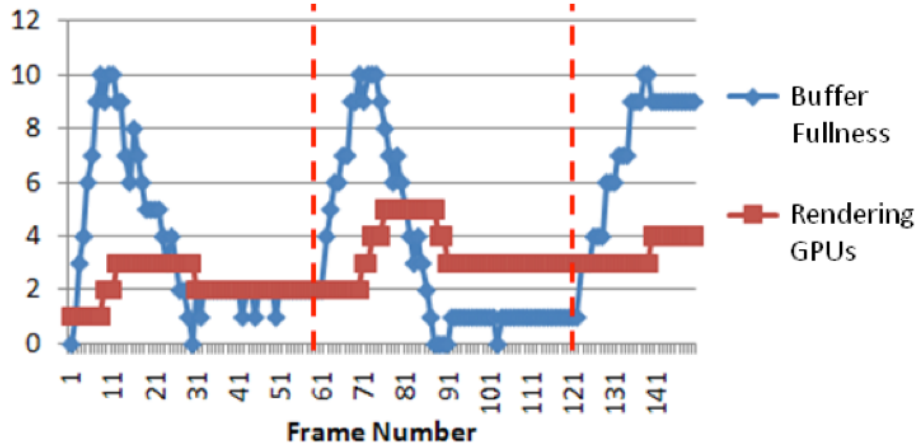
$$\psi_i^{sim} - \psi_i^{vis} \approx \phi(p) = \phi(F) = \begin{cases} -1.0 & \text{if } F = 0, \\ 0.0 & \text{if } F \in \{1, \dots, B-1\}, \\ 1.0 & \text{if } F = B, \end{cases}$$

Dynamic Load Balancing: Result

- **Stability of the Dynamic Scheduling Algorithm**

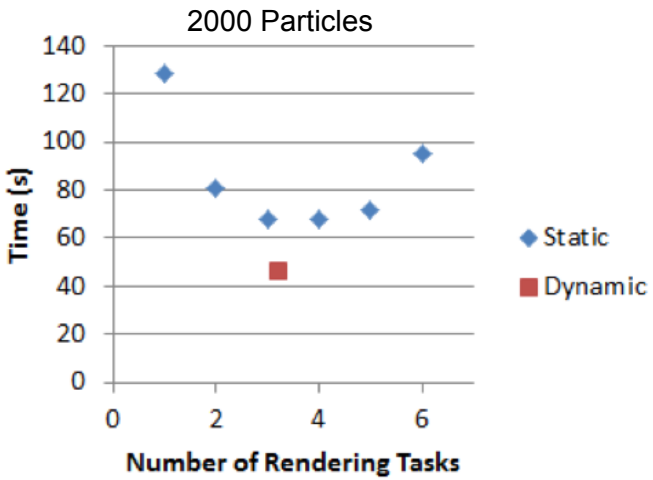


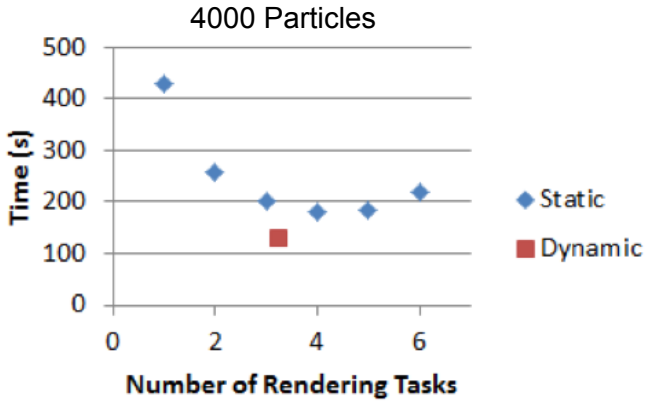
No parameter change (only at the beginning)

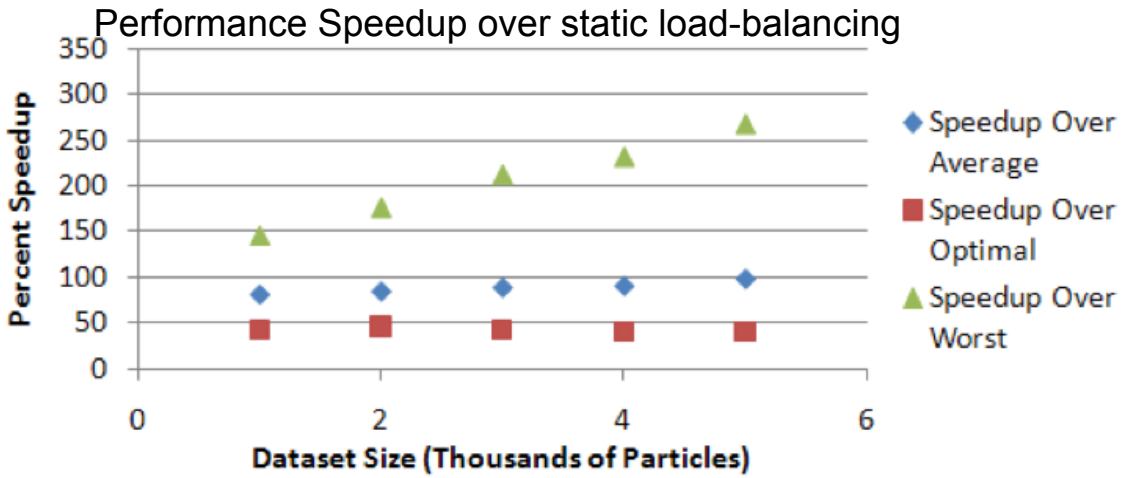


Parameters change at the dotted line.

Comparison: Dynamic vs. Static Scheduling







Conclusion

+ Pipelining
+ Dynamic load balancing
- Fine granularity load balancing (SM level)
- Communication overhead
- Programmability: Software framework, Library

Question(s):

- **Contact Information:**

Yong Cao
Computer Science Department
Virginia Tech
Email: yongcao@vt.edu
Website: www.cs.vt.edu/~yongcao