Proximity Computations on Heterogeneous Computing Systems

Duksu Kim
Ph. D. candidate, Scalable Graphics Lab.,
Korea Advanced Institute of Science and Technology (KAIST)
Proximity Queries (PQs)

- Compute relative placement or configuration of two objects
  - Collision detection
  - Distance computation

- Basic operations in various applications
  - Graphics, simulations, robotics, Etc.
Proximity Queries in App.

- Motion planning
  - Collision detection
  - Others

- Realistic rendering
  - Ray tracing
  - Others

References:
- [Jia 2010]
- [Liangjun 2008]

From Bochang's paper
Proximity Query Acceleration

- Various acceleration techniques
  - Acceleration hierarchies
  - Culling algorithms
  - Specialize algorithms for a target application
  - Approximation algorithms

- Achieve several orders of magnitude performance improvement
Proximity Query Acceleration

• Not enough to achieve real-time performance for large-scale models and complex scenes

Continuous collision detection

N-body benchmark consisting of 34K triangles

Less than 10 frames/second (Intel i7 2.93Ghz CPU)
Demands for High Performance

- Model complexity continue to grow for more realistic and accurate outputs
- Applications require interactive/real-time performance

from Creative Assembly’s “Rome: Total war”
Parallel Computing Trend

- Multi/Many-core architectures
  - Multi-core CPU
  - Graphics processing unit (GPU)
Parallel Computing Trend

- **Multi/Many-core architectures**
- **Heterogeneous architectures**
  - Different types of computing devices in a system
    - Multi-core CPUs and GPUs in a PC
    - Intel Sandy Bridge, AMD Fusion, Sony Cell, ..
Related Work

- Multi-core CPU-based approaches
  - Metric-based load-balancing method [Lee 2010]
  - Front based task decomposition method [Tang 2009]
  - Parallel BVH construction [Wald 2007] [Ize 2007]
  - Voxel-based method [Lawlor 2002]

- GPU-based proximity query algorithms
  - Visibility queries [Govindaraju 2005]
  - Image-based approach [Govindaraju 2005]
  - Unified GPU-framework for proximity queries [Sud 2006] [Lauterbach 2010]
  - Specialized on certain types of models [Vassilev 2001] [Baciu 2002] [Govindaraju 2005*]

- Achieve high performance improvement
- Use only multi-core CPUs or GPUs
- Do not provide real-time performance yet for large-scale models
Our Goals & Approaches

• Achieve real-time performance
  – In various proximity queries for large-scale models

• Efficiently utilize all available computing resources for proximity computations
  – Both GPUs and multi-core CPUs

• Design an optimization-based scheduling (work distribution) algorithm
Previous Work:

Hierarchical Jobs

Non-hierarchical Jobs

HPCCD: Hybrid Parallel Continuous Collision Detection
Previous Work:

Hierarchical Jobs
- Branch prediction
- Cache

Non-hierarchical Jobs
- Random accesses
- Solving cubic equations

HPCCD:
Hybrid Parallel Continuous Collision Detection
- Massive parallelism
• Manually specify distribution rules depending on knowledge on the application

* This work was published at Computer Graphics Forum 2009 (received the distinguished paper award form Pacific Graphics 2009)
Limitations

- Manually specify distribution rules depending on knowledge on the application
- No guarantee to efficient utilization of computing resources
Current Work

• Previous work: Manual scheduling
  – Application dependent heuristics
  – No guarantee to optimality
Current Work

• Previous work: Manual scheduling

• Optimization-based scheduling
  – Automatically distribute dynamically generated jobs while considering the optimal utilization of computing resources
Current Work – Results

(With our optimization-based scheduling)

Collision detection
Fracturing benchmark (252K Tri.)

- Work stealing
  (an well-known algorithm for dynamic workload distribution)

- Use same GPUs (low heterogeneity)

- Results
  FPS

- GTX480 + GTX480 + GTX480 + GTX580

Use same GPUs (low heterogeneity)
Current Work – Results
(With our optimization-based scheduling)

Use different GPUs (high heterogeneity)

Collision detection
Fracturing benchmark (252K Tri.)

Work stealing

Ours

FPS

Use different GPUs (high heterogeneity)
Outline

• Motivation
• Our approach
  – Optimization-based scheduling
• Results
• Conclusion
Overview

Proximity query

Hierarchical

Leaf-level computation

Optimization based job scheduling

Resource

Resource

Resource
Overview

Proximity query

Hierarchical

Leaf-level computation

Optimization based job scheduling

Resource

Resource

Resource
Related Work: Scheduling

- **Scheduling for homogeneous resources**
  - Do not consider properties of heterogeneous computing environments

- **Application-dependent heuristics**
  - Unclear how well these techniques can be applied to other applications

- **Optimization-based scheduling**
  - Compute optimal job distribution that minimize computation time
Related Work: Scheduling

• Scheduling for homogeneous resources
  – Do not consider properties of heterogeneous computing environments

• Application-dependent heuristics
  – Unclear how well these techniques can be applied to other applications

• Optimization-based scheduling
  – Compute optimal job distribution that minimize computation time
Related Work: Scheduling

• **Scheduling for homogeneous resources**
  – Do not consider properties of heterogeneous computing environments

• **Application-dependent heuristics**
  – Unclear how well these techniques can be applied to other applications

• **Optimization-based scheduling**
  – Compute optimal job distribution that minimize computation time
Outline

• Motivation
• Our approach
  – Optimization-based scheduling
• Results
• Conclusion
Optimization-based Scheduling

- Design an accurate performance model
  - Predict how much computation time is required to finish jobs on a resource
  - Important to achieve the optimal scheduling result
Performance Model

- Complex relationship between jobs and resources
Performance Model

- Abstract the complex relationship as an expected running time model

Job type 1

... Expected Running Time

Job type n

Resource 1
- Processor architecture
- Memory
- Execution model
- Communication
... Resource 2
- Processor architecture
- Memory
- Execution model
- Communication
...
• Running time is **linearly increased** as the number of jobs is increased
• Running time is **linearly increased** as the number of jobs is increased

• Each computing resource require a specific amount of **setup cost**
Performance Model

- Inter-device data transfer time depends on the pair of devices
- Data transfer time is linearly increased as the number of jobs is increased
Expected Running Time Model

- Expected running time on computing resource $i$ for processing $n$ jobs of job types $j$ that are generated from computing resource $k$

$$T(k \rightarrow i, j, n_{ij}) = \begin{cases} 
0, & \text{if } n_{ij} \text{ is 0} \\
T_{\text{setup}}(i, j) + T_{\text{proc}}(i, j) \times n_{ij} + T_{\text{trans}}(k \rightarrow i, j) \times n_{ij}, & \text{otherwise.}
\end{cases}$$
Expected Running Time Model

• Measure coefficients of our linear formulation for each proximity query with sample jobs

  – The expected running time model shows high correlation (0.91 on average) with the observed data in tested benchmarks

\[
T(k \rightarrow i, j, n_{ij}) = \begin{cases} 
0, & \text{if } n_{ij} \text{ is 0} \\
T_{\text{setup}}(i, j) + T_{\text{proc}}(i, j) \times n_{ij} \\
+ T_{\text{trans}}(k \rightarrow i, j) \times n_{ij}, & \text{otherwise.}
\end{cases}
\]
Optimization-based Scheduling

Expected running time model

Optimization formulation

Iterative LP solver

• Formulate an optimization problem
  – Based on the expected running time model
  – Need to represent the scheduling problem as a form of optimization problem

\[
T(k \rightarrow i, j, n_{ij}) = \begin{cases} 
0, & \text{if } n_{ij} \text{ is 0} \\
T_{\text{setup}}(i, j) + T_{\text{proc}}(i, j) \times n_{ij} & \\
+ T_{\text{trans}}(k \rightarrow i, j) \times n_{ij}, & \text{otherwise.}
\end{cases}
\]
Optimization Formulation

- Minimize makespan problem

\[ \text{Minimize } L, \]

Computing resource

- CPU 1
- CPU 2
- GPU 1
- GPU 2

Processing time

Time

Makespan \((L)\)
Optimization Formulation

- We calculate optimal job distribution with the expected running time

Minimize $L$, subject to $T_{rest}(i) + \sum_{j=1}^{\lfloor J \rfloor} T(i, j, n_{ij}) \leq L, \forall i \in R$ \small{①}

① The expected processing time of computing resources is equal or smaller than the makespan

Resource $i$ | Expected processing time | Rest time for completing already assigned jobs | Processing time for jobs will be assigned | Makespan
Optimization Formulation

- We calculate optimal job distribution with the expected running time

\[
\text{Minimize } L,
\]

subject to

\[
T_{\text{rest}}(i) + \sum_{j=1}^{|J|} T(i, j, n_{ij}) \leq L, \forall i \in R \\
\sum_{i=1}^{|R|} n_{ij} = n_j, \forall j \in J
\]

1. The expected processing time of computing resources is equal or smaller than the makespan
2. There is no missing or duplicated jobs
We calculate optimal job distribution with the expected running time.

Minimize $L,$

subject to $T_{rest}(i) + \sum_{j=1}^{J} T(i, j, n_{ij}) \leq L, \forall i \in R$

$\sum_{i=1}^{R} n_{ij} = n_j, \forall j \in J$

$n_{ij} \in \mathbb{Z}^+(zero\ or\ positive\ integers).$

1. The expected running processing of computing resources is equal or smaller than the makespan.
2. There is no missing or duplicated jobs.
3. Each job is atomic.
Minimize $L$, subject to $T_{rest}(i) + \sum_{j=1}^{J} T(i, j, n_{ij}) \leq L, \forall i \in R$

$\sum_{i=1}^{R} n_{ij} = n_j, \forall j \in J$

$n_{ij} \in \mathbb{Z}^+(\text{zero or positive integers})$. 
Optimization-based Scheduling

Expected running time model $\rightarrow$ Optimization formulation $\rightarrow$ Iterative LP solver

Minimize $L$,
subject to $T_{rest}(i) + \sum_{j=1}^{J} T(i, j, n_{ij}) \leq L, \forall i \in R$
$\sum_{i=1}^{R} n_{ij} = n_j, \forall j \in J$
$n_{ij} \in \mathbb{Z}^+(\text{zero or positive integers})$.

**NP-hard Problem!**

- High computational cost
  - Jobs are dynamically generated at runtime
  - Optimization process takes long time to apply to interactive or real-time applications
Optimization-based Scheduling

Expected running time model

\[ T(k \rightarrow i, j, n_{ij}) = \begin{cases} 0, & \text{if } n_{ij} \text{ is 0} \\ T_{\text{setup}}(i, j) + T_{\text{proc}}(i, j) \times n_{ij} \\ + T_{\text{trans}}(k \rightarrow i, j) \times n_{ij}, & \text{otherwise.} \end{cases} \]

Optimization formulation

\[
\text{Minimize } L, \\
\text{subject to } T_{\text{rest}}(i) + \sum_{j=1}^{\left|J\right|} T(i, j, n_{ij}) \leq L, \forall i \in R \\
\sum_{i=1}^{\left|R\right|} n_{ij} = n_j, \forall j \in J \\
n_{ij} \in \mathbb{Z}^+(\text{zero or positive integers}).
\]

Iterative LP solver

Designed iterative solve to handle the piece-wise condition

Positive floating-point numbers
Optimization-based Scheduling

Expected running time model → Optimization formulation → Iterative LP solver

Please see the technical report for the details (http://sglab.kaist.ac.kr/Hybrid_parallel)

\[
\begin{align*}
\text{subject to } & T_{\text{rest}}(i) + \sum_{j=1}^{\lvert J \rvert} T(i, j, n_{ij}) \leq L, \forall i \in R \\
& \sum_{i=1}^{\lvert R \rvert} n_{ij} = n_j, \forall j \in J \\
& n_{ij} \in \mathbb{Z}^+ (\text{zero or positive integers}).
\end{align*}
\]
• Motivation
• Our approach
  – Optimization-based scheduling
• Results
• Conclusion
Results

• Applied to various application
  – Collision detection
  – Motion planning
  – Global illumination
Results

Collision detection (146K Tri.)

- FPS
- Tesla 2075
- GTX 480
- GTX 580
- Hex-core CPUs (Intel Xeon (2.93GHz))

Use different GPUs

- GTX 285
- Work steal
- Round-robin

Graph showing performance with different GPU configurations.
Results

Motion planning
(137K Tri., 50K samples)
- FPS
- Work stealing
- Round-robin

Global illumination
(436K Tri., 80M rays)
- FPS
- Work stealing
- Round-robin

Ours

Use different GPUs
Motivation

Our approach
  – Optimization-based scheduling

Results

Conclusion
Conclusion

• **Present a novel scheduling algorithm**
  – Design the expected running time model
  – Formulate the scheduling problem as an optimization problem
  – Propose a novel iterative optimization solver

• **Efficiently utilize heterogeneous computing systems**
  – Achieve high scalability with additional computing resources
  – In various proximity queries
Future Work

- Apply to other applications
- Design a better scheduling algorithm
References


• [Lauterbach 2010] gProximity: Hierarchical GPU-based Operations for Collision and Distance Queries, C Lauterbach et al., EG 2010


Thanks

Any questions?

(bluekdct@gmail.com)

Project homepage:
http://sglab.kaist.ac.kr/hybrid_parallel

* This work was submitted to a journal and under a minor revision.