Scheduling in Heterogeneous Computing Environments for Proximity Queries

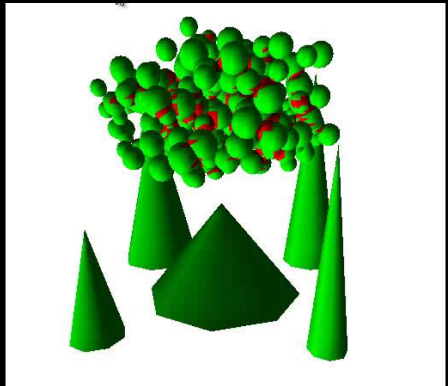
IEEE TVCG, Sept., 2013

Presenter:
Duksu Kim Jinkyu Lee Junghwan Lee
Insik Shin John Kim Sung-Eui Yoon
KAIST (Korea Advanced Institute of Science and Technology)

This presentation slides are available at http://sglab.kaist.ac.kr/Hybrid_parallel

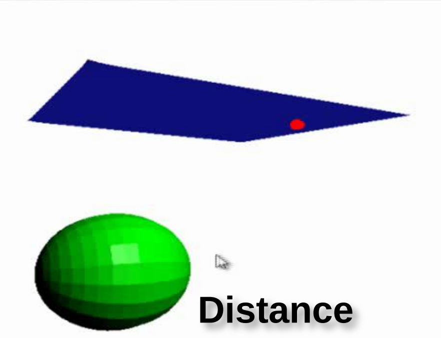
Proximity Queries (PQs)

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



Collision detection

- **Basic operations in various applications**
  - Graphics, simulations, robotics, Etc.

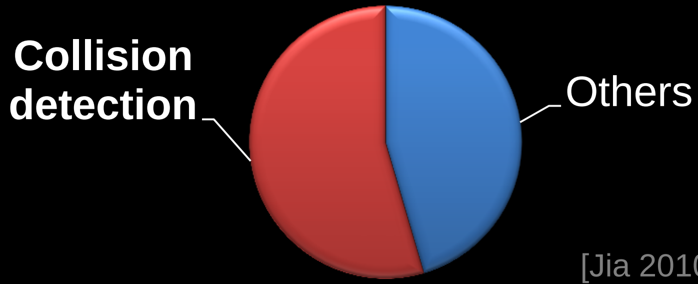


Distance computation

Proximity Queries in Applications

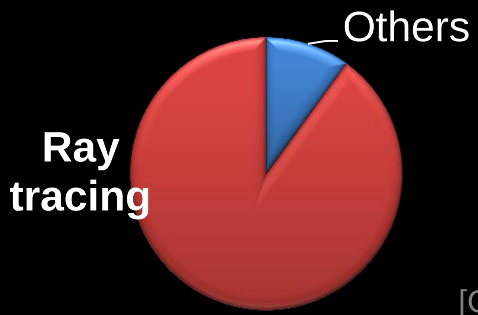
from Moon et al. 2009

Motion planning



[Jia 2010] [Liangjun 2008]

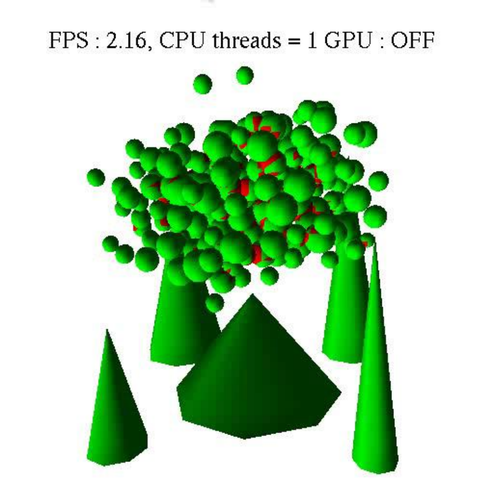
Realistic redering



[Our in-house render]

Proximity Query Acceleration

- **Various prior acceleration techniques have been proposed (e.g., culling algorithms)**
- **Not enough to achieve real-time performance for large-scale models**



Continuous collision detection

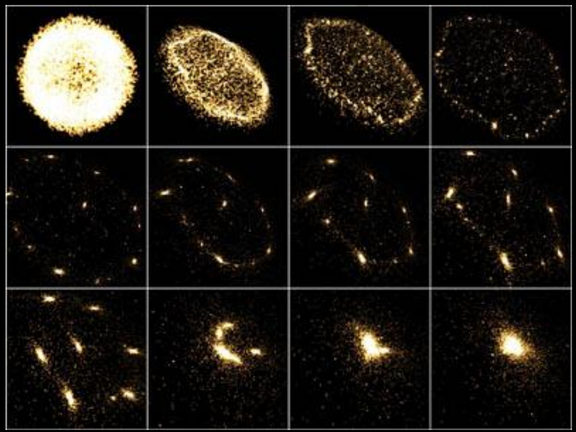
N-body benchmark
consisting of 34K triangles

Less than 10 frames/second
(Intel i7 2.93Ghz CPU)

Demands for High Performance

- Model complexity continues to grow for more realistic and accurate results
- Applications require real-time performance

from Creative Assembly's "Rome: Total war"



N-body simulation from NVIDIA

Parallel Computing Trend

- **Multi/many-core architectures**
  - Multi-core CPU
  - Graphics processing unit (GPU)

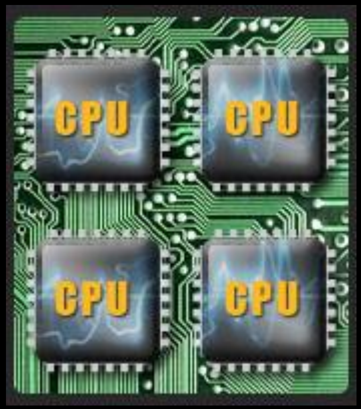


Image from NVIDIA

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, ..

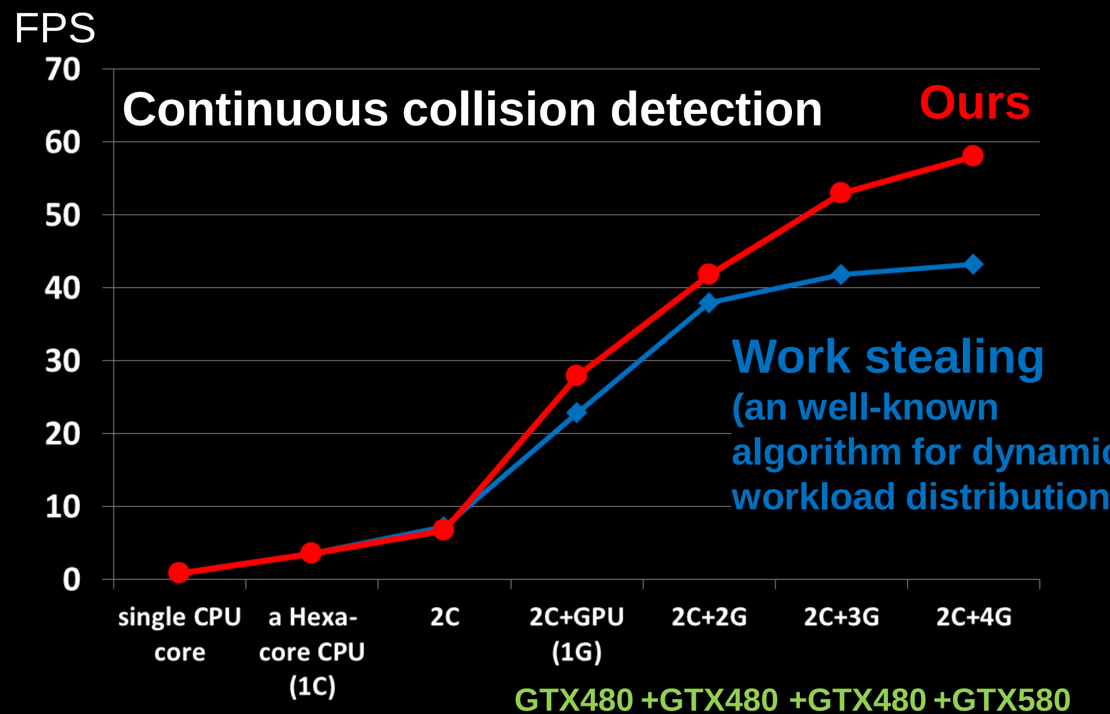
Images from NVIDIA

Our Goal & 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**

Current Work – Results

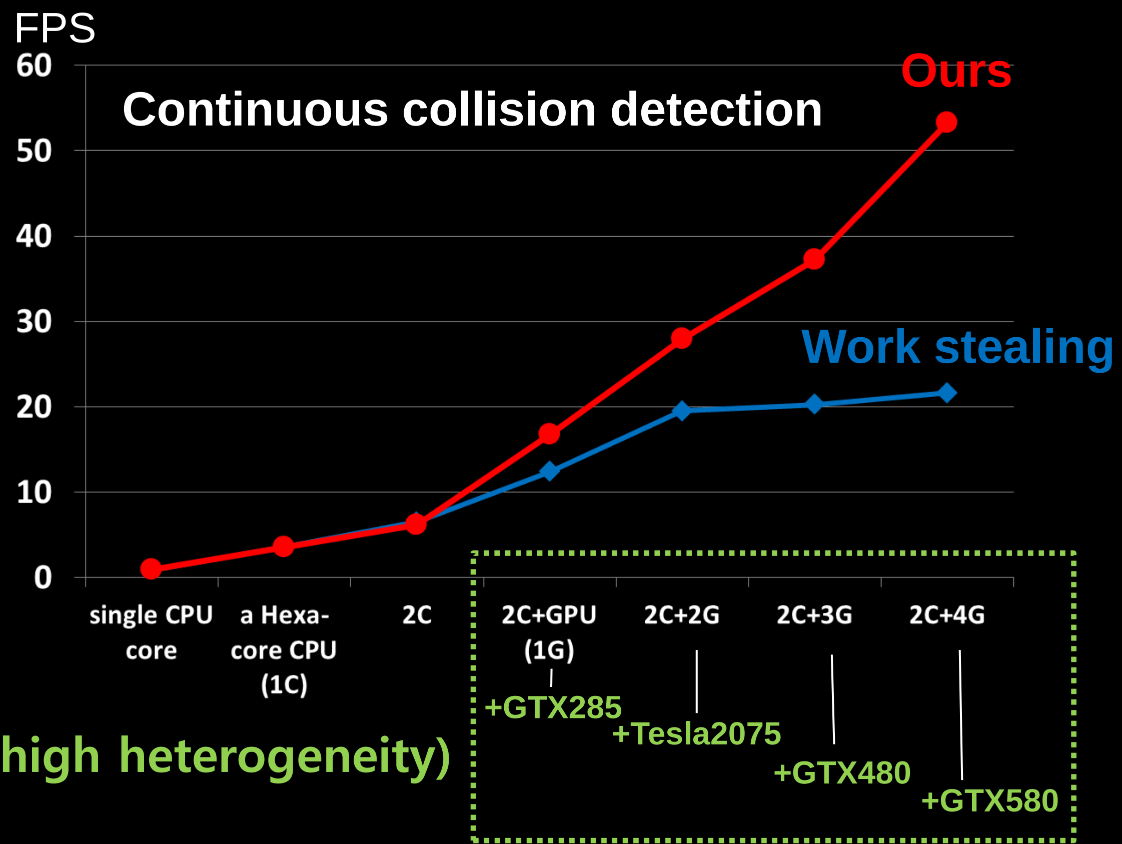
(With our optimization-based scheduling)



Use same GPUs (low heterogeneity)

Current Work – Results

(With our optimization-based scheduling)



Use different GPUs (high heterogeneity)

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*]

Related Work

- **Multi-core CPU-based approaches**
  – Metric-based load-balancing method [Lee 2010]

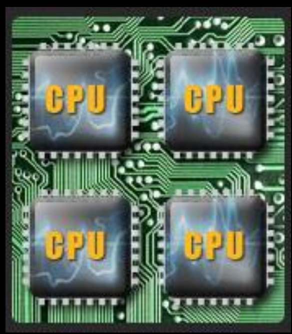
- **Achieve high performance improvement**
- **Use only multi-core CPUs or GPUs**
- **Do not provide real-time performance yet for large-scale models**

  – 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*]

Related Work

- **Utilize both multi-core CPUs and GPUs**
  - **HPCCD**: Hybrid Parallel Continuous Collision Detection [Kim 2009]
    - Our previous work

Related Work: HPCCD



- Branch prediction
- Cache

Hierarchical Jobs

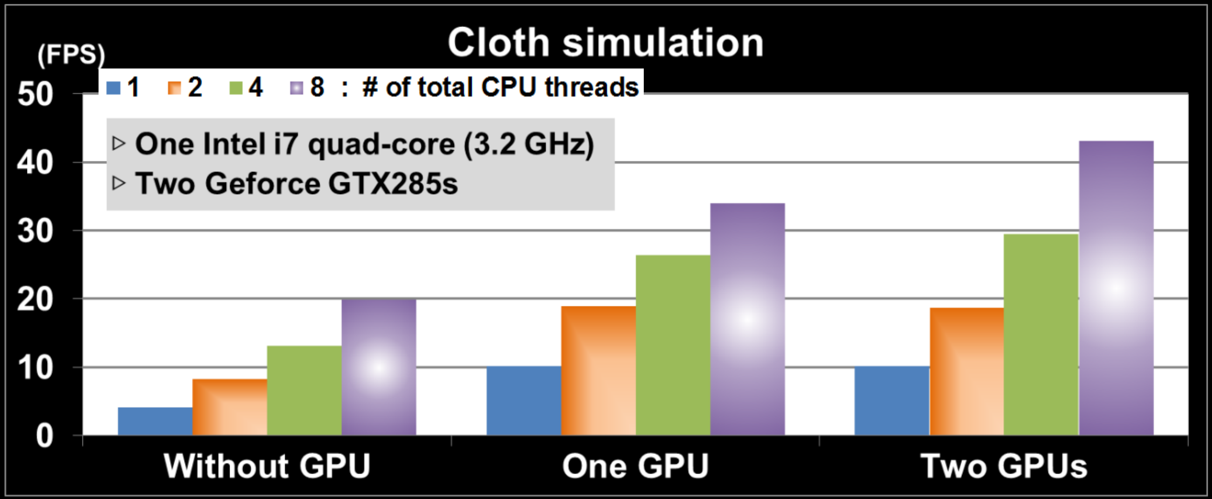
Random accesses

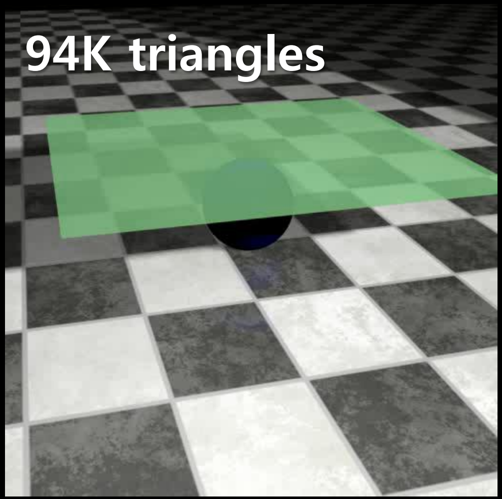
Non-hierarchical Jobs

Solving cubic equations

- Massive parallelism

Related Work: HPCCD

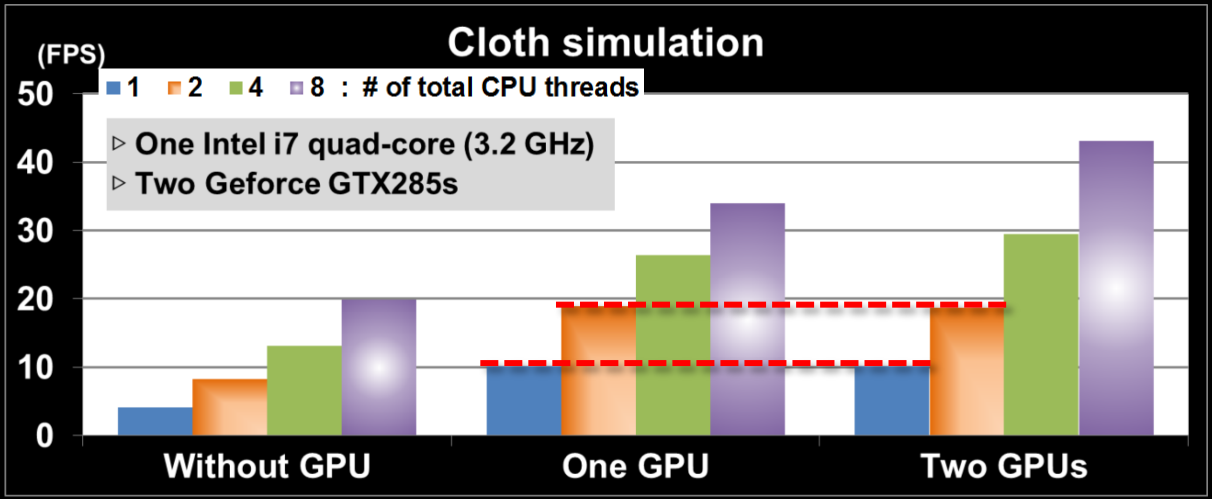


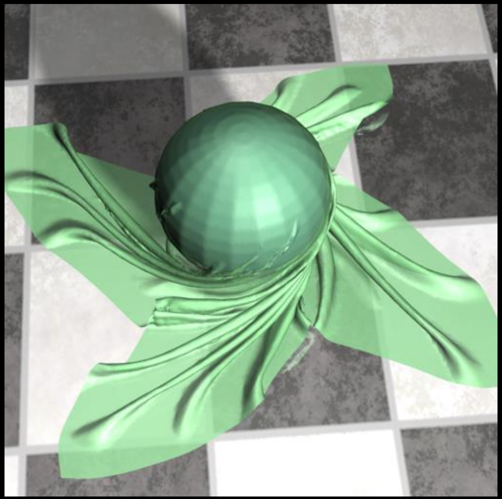


- **Manually specify distribution rules depending on the knowledge on the application**

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

Related Work: HPCCD





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

Related Work: Scheduling

- **Application-dependent heuristics (e.g., HPCCD)**
  - Unclear how well these techniques can be applied to other applications
- **Scheduling for homogeneous resources**
  - Do not consider properties of heterogeneous computing environments
- **Optimization-based scheduling**
  - Designed for general problems
  - Compute the optimal job distribution that minimizes computation time

Related Work: Scheduling

- **Application-dependent heuristics (e.g., HPCCD)**
  - Unclear how well these techniques can be applied to other applications
- **Scheduling for homogeneous resources**
  - Do not consider properties of heterogeneous computing environments
- **Optimization-based scheduling**
  - Designed for general problems
  - Compute the optimal job distribution that minimizes computation time

Our Research Direction

- **Previous work: Manual scheduling**
  - Application dependent heuristics
  - No guarantee to optimality

Our Research Direction

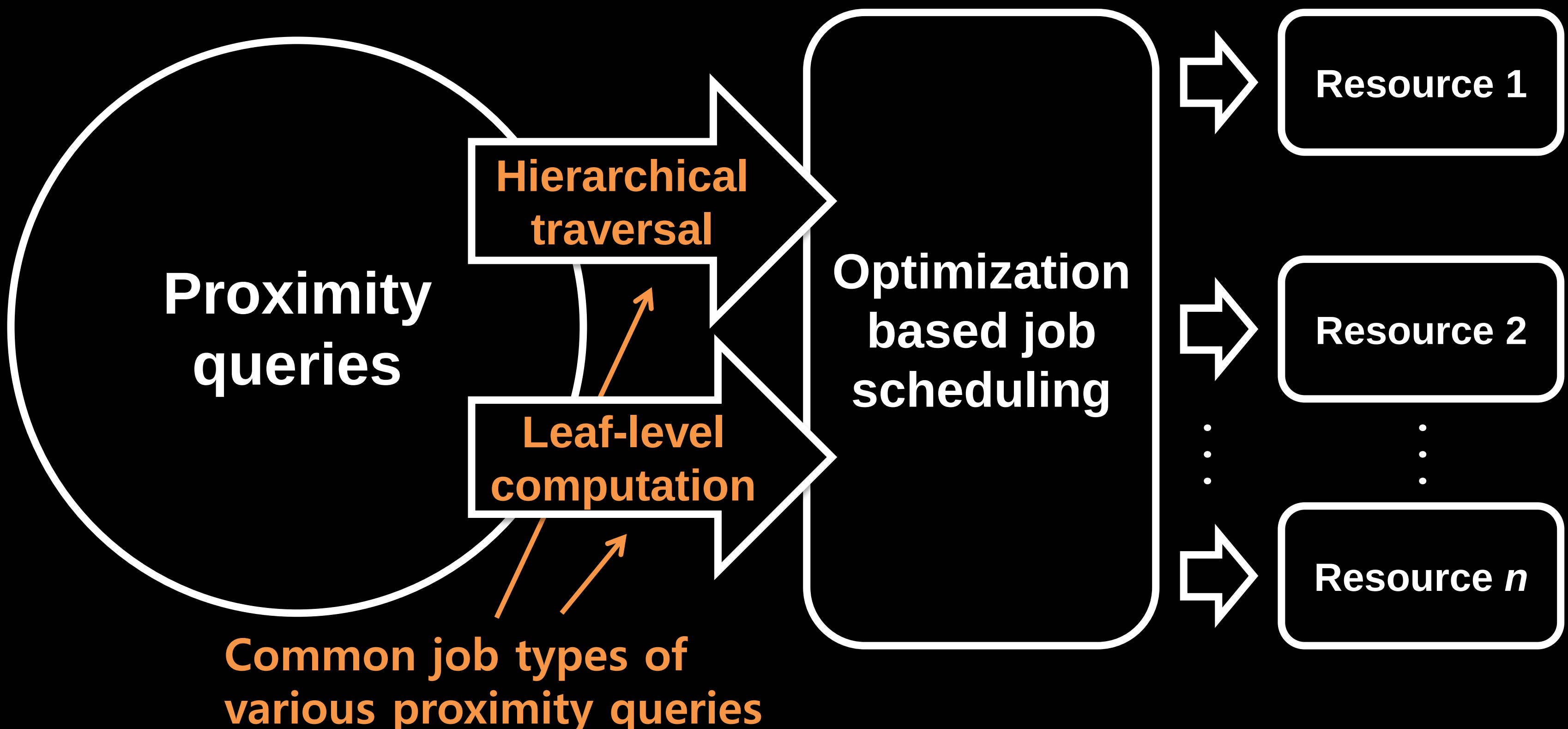
- Previous work: Manual scheduling

- Optimization-based scheduling
  - Automatically distribute dynamically generated jobs, while considering the optimal utilization of computing resources

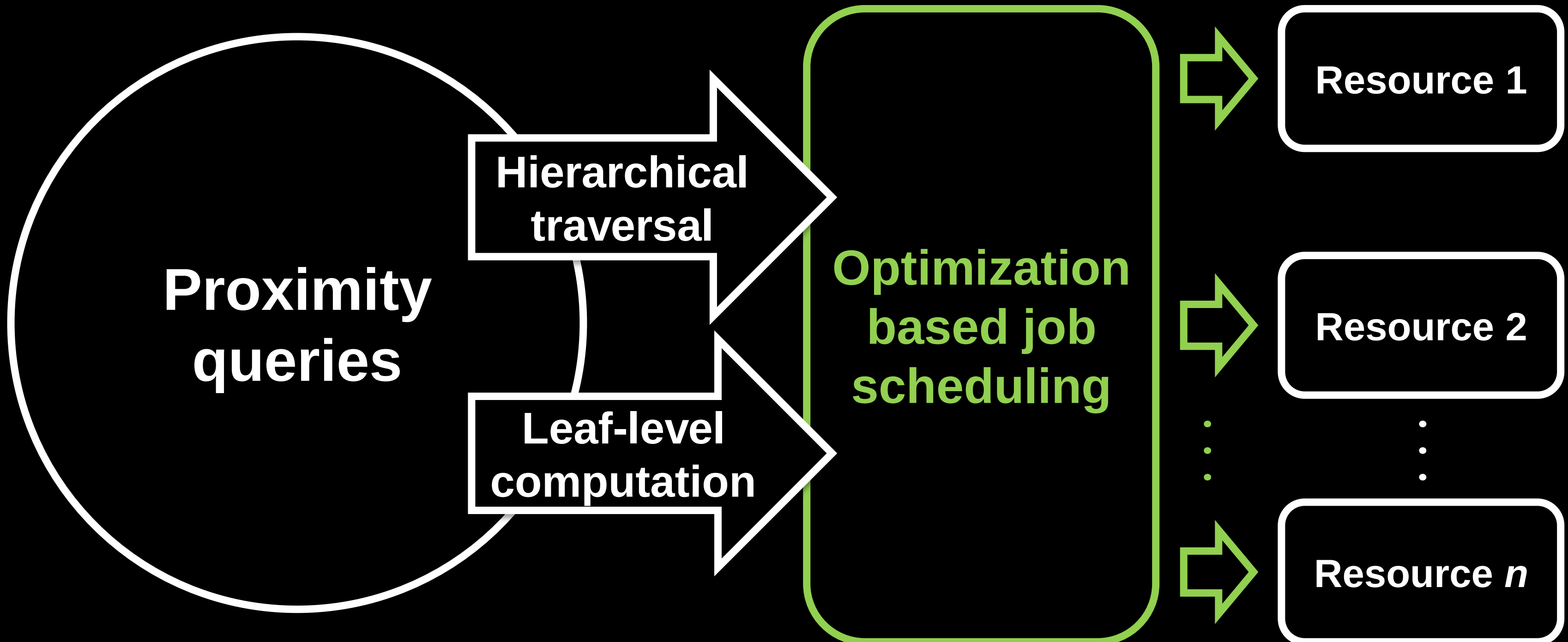
Outline

- Motivation
- Our approach
  - Optimization-based scheduling
- Results
- Conclusion

Overview


Proximity queries
Hierarchical traversal
Leaf-level computation
Optimization based job scheduling
Resource 1
Resource 2
Resource *n*
Common job types of various proximity queries

Overview


Proximity queries
Hierarchical traversal
Leaf-level computation
Optimization based job scheduling
Resource 1
Resource 2
Resource *n*

Outline

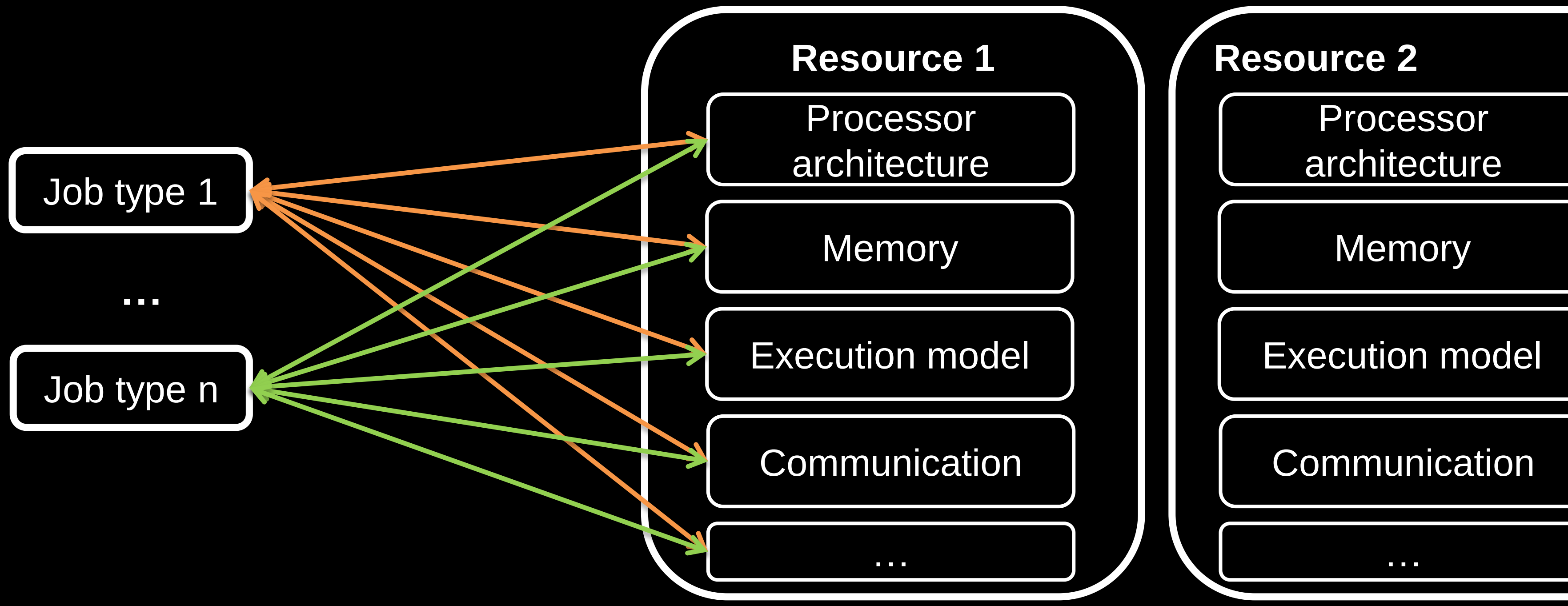
- Motivation
- Our approach
  - Optimization-based scheduling
- Results
- Conclusion

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

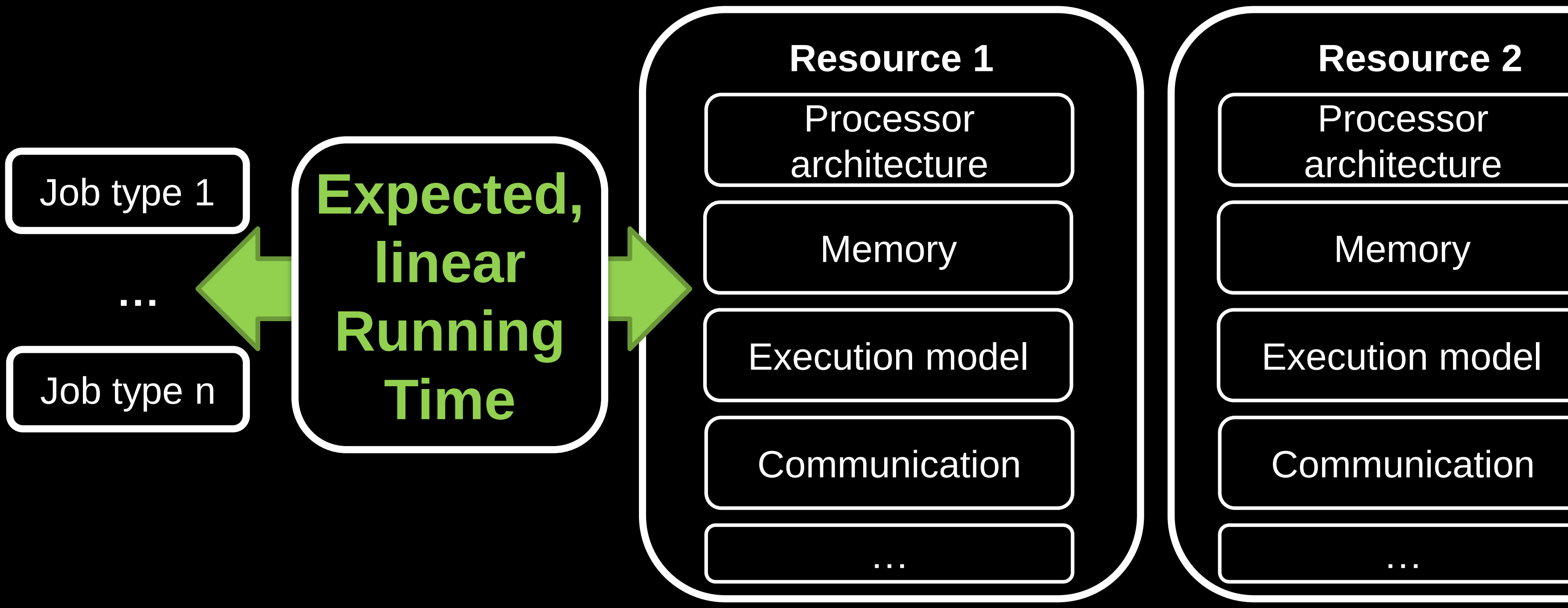
Performance Model

- **Performance relationship between jobs and resources is complex**

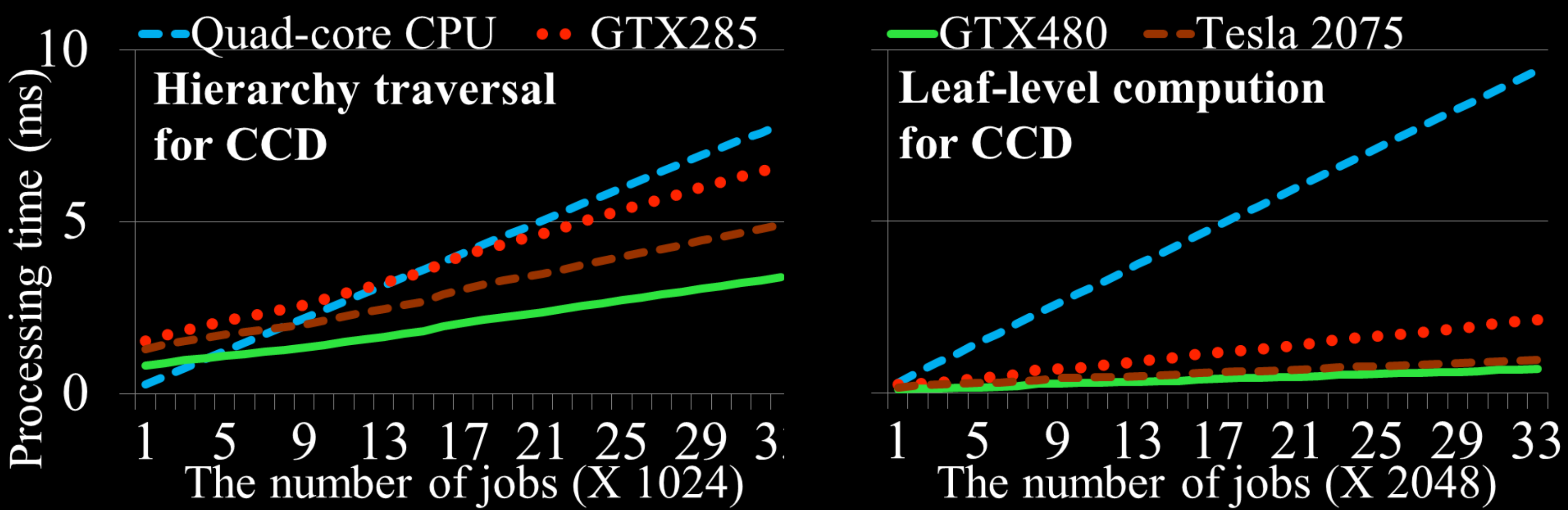


Performance Model

- **Abstract the complex relationship as an expected, linear model**

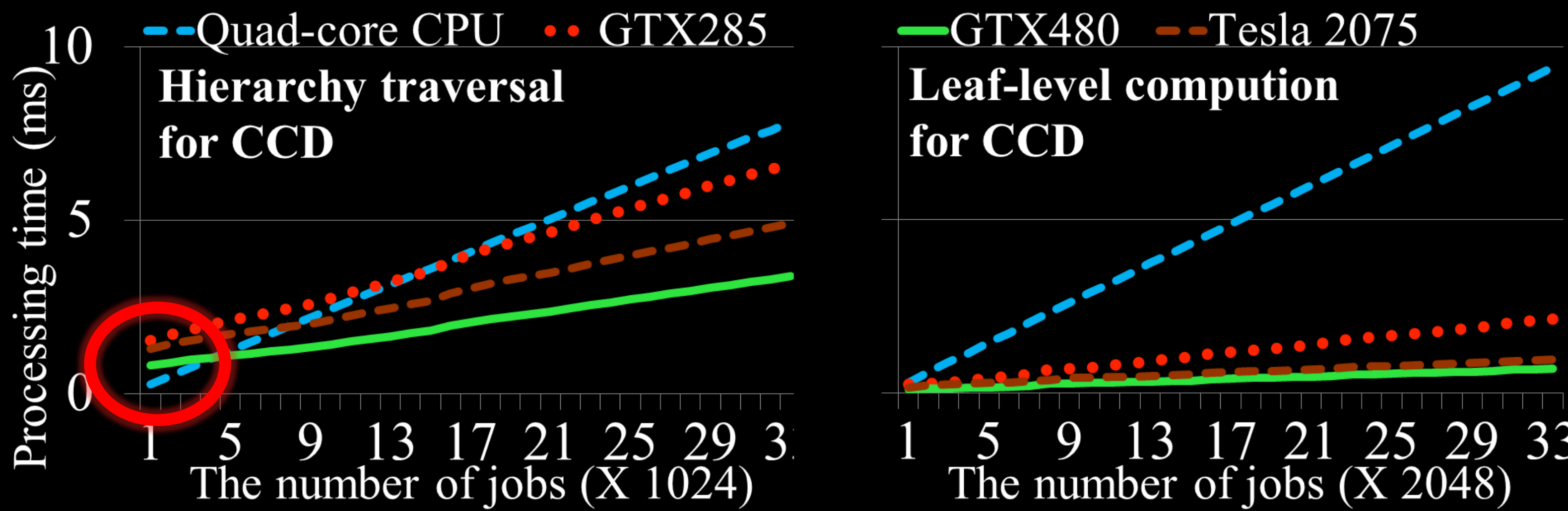


Performance Model



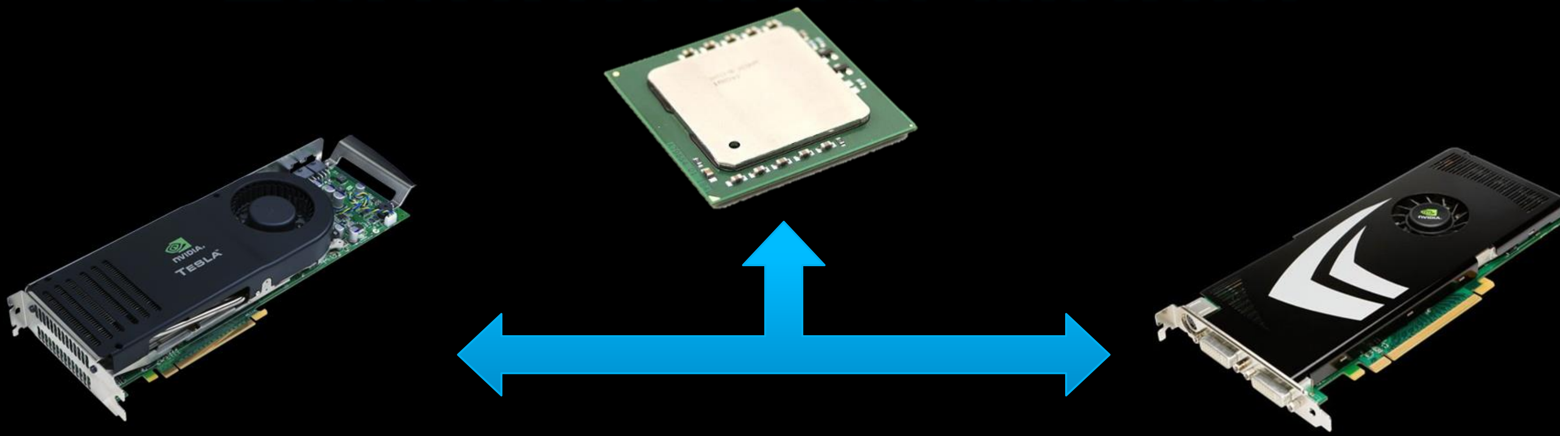
- **Running time is linearly increased as the number of jobs is increased**

Performance Model



- Running time is linearly increased as the number of jobs is increased
- Each computing resource requires 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

- T() : Expected running time on computing resource *i* for processing *n* jobs of job types *j* that are generated from computing resource *k*

$$T(k \to i, j, n_{ij}) = \begin{cases} 0, & \text{if } n_{ij} \text{ is } 0 \\ \underbrace{T_{setup}(i,j)}_{\text{Setup time}} + \underbrace{T_{proc}(i,j) \times n_{ij}}_{\text{Processing time}} & \\ \underbrace{+T_{trans}(k \to i, j) \times n_{ij}}_{\text{Data transfer time}}, & \text{otherwise.} \end{cases}$$

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

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

Optimization-based Scheduling

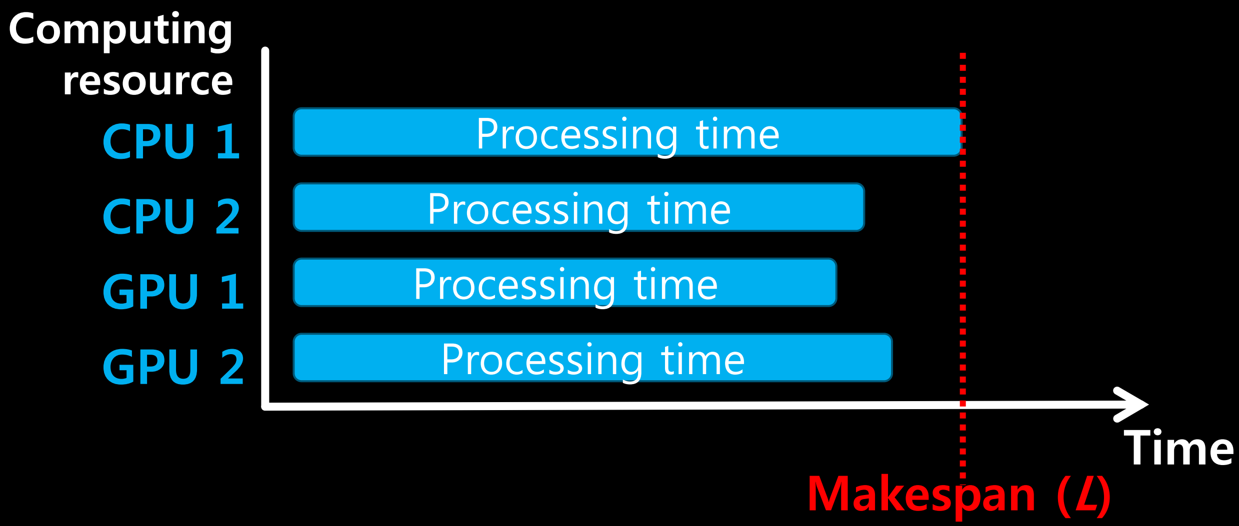
$$T(k \to i, j, n_{ij}) = \begin{cases} 0, & \text{if } n_{ij} \text{ is } 0 \\ T_{setup}(i,j) + T_{proc}(i,j) \times n_{ij} & \\ \quad + T_{trans}(k \to i, j) \times n_{ij}, & \text{otherwise.} \end{cases}$$

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

Optimization Formulation

- **Minimize the makespan (L) problem**

$$Minimize\ L,$$



Optimization Formulation

- **Calculate the optimal job distribution with the expected running time**

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

Expected processing time

Resource i

Rest time for completing already assigned jobs

Processing time for jobs will be assigned

Makespan

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

Optimization Formulation

- **We calculate optimal job distribution with the expected running time**

$$Minimize\ L,$$

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

$$\Sigma_{i=1}^{|R|} n_{ij} = n_j, \forall j \in J \quad ②$$

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

② There are no missing or duplicated jobs

Optimization Formulation

- **We calculate optimal job distribution with the expected running time**

Job distribution

$$Minimize\ L,$$

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

$$\Sigma_{i=1}^{|R|} n_{ij} = n_j, \forall j \in J \quad ②$$

$$n_{ij} \in \mathbb{Z}^{+}(zero\ or\ positive\ integers). \quad ③$$

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

② There are no missing or duplicated jobs

③ Each job is atomic

Optimization-based Scheduling

Expected running time model → Optimization formulation → Iterative LP solver

$$\begin{aligned}
&Minimize\ L,\\
&subject\ to\ T_{rest}(i) + \Sigma_{j=1}^{|J|} T(i,j,n_{ij}) \leq L, \forall i \in R\\
&\qquad\qquad \Sigma_{i=1}^{|R|} n_{ij} = n_j, \forall j \in J\\
&\qquad\qquad n_{ij} \in \mathbb{Z}^{+} (zero\ or\ positive\ integers).
\end{aligned}$$

Optimization-based Scheduling

Expected running time model → **Optimization formulation** → **Iterative LP solver**

$$\begin{aligned} &Minimize\ L, \\ &subject\ to\ T_{rest}(i) + \Sigma_{j=1}^{|J|} T(i,j,n_{ij}) \le L, \forall i \in R \\ &\qquad \Sigma_{i=1}^{|R|} n_{ij} = n_j, \forall j \in J \\ &\qquad n_{ij} \in \mathbb{Z}^+ (zero\ or\ positive\ integers). \end{aligned}$$

NP-hard Problem!

- **High computational cost**
  - Jobs are dynamically generated at runtime
  - Optimization process takes long time for interactive or real-time applications

Optimization-based Scheduling

Expected running time model → **Optimization formulation** → **Iterative LP solver**

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

Designed an iterative LP solving algorithm to handle the piece-wise condition

$$\begin{aligned} &Minimize\ L, \\ &subject\ to\ T_{rest}(i) + \Sigma_{j=1}^{|J|} T(i, j, n_{ij}) \leq L, \forall i \in R \\ &\qquad\qquad \Sigma_{i=1}^{|R|} n_{ij} = n_j, \forall j \in J \\ &\qquad\qquad n_{ij} \in \mathbb{Z}^{+}(zero\ or\ \cancel{positive\ integers}). \end{aligned}$$

Positive floating-point numbers

Optimization-based Scheduling

Expected running time model → **Optimization formulation** → **Iterative LP solver**

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

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

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

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

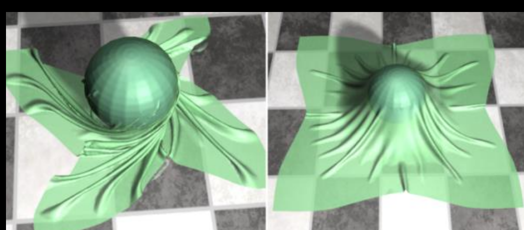
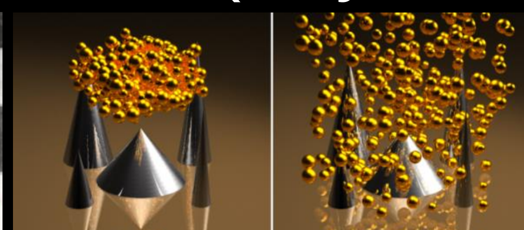
Positive floating-point numbers

Outline

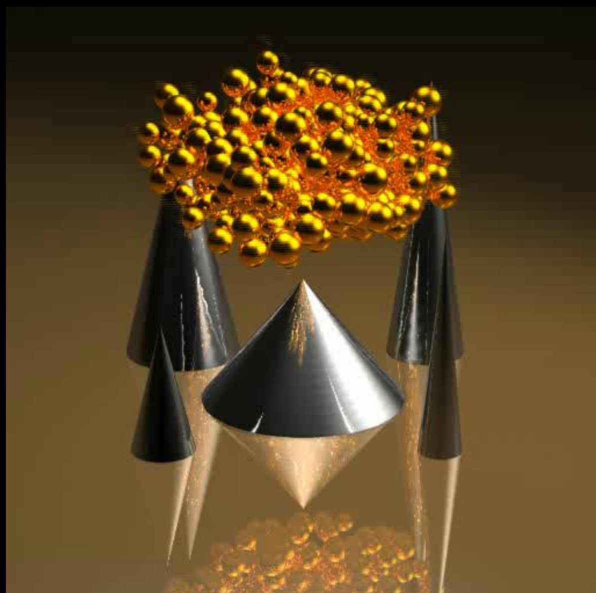
- **Motivation**
- **Our approach**
  - Optimization-based scheduling
- **Results**
- **Conclusion**

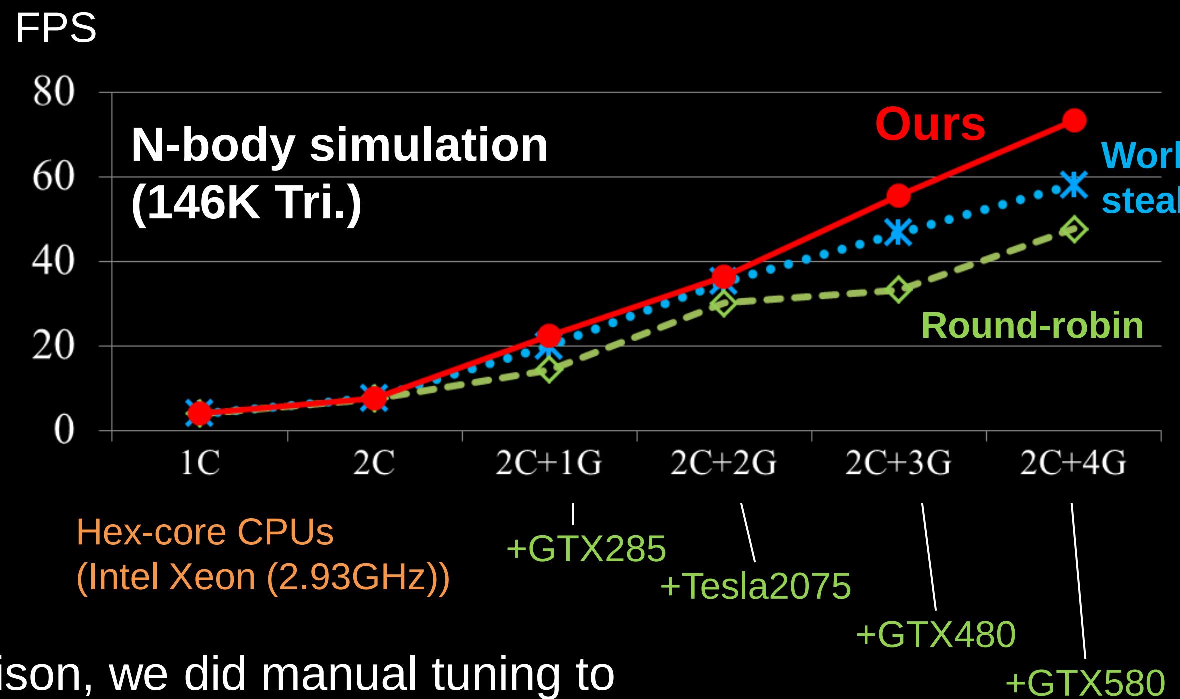
Results

- **Tested with various applications**
  - Simulations (Continuous collision detection)
  - Motion planning (Discrete collision detection)
  - Global illumination (Ray-Triangle intersection)


Results

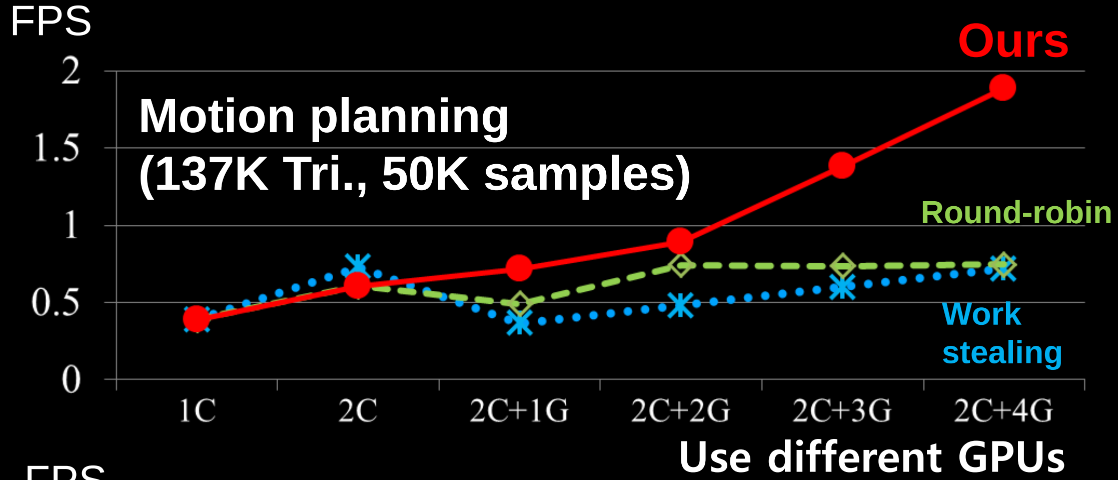


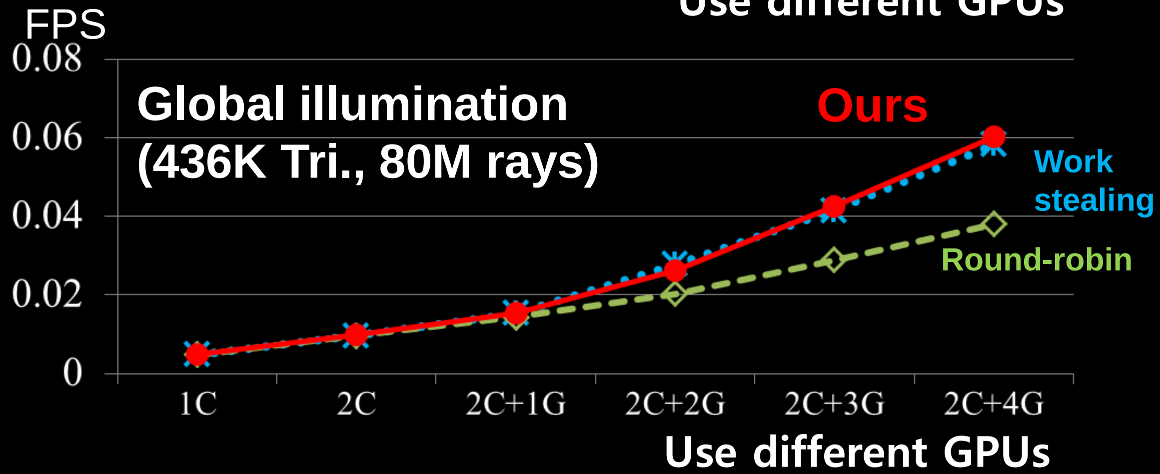


Use different GPUs

- For conservative comparison, we did manual tuning to get the best performance for tested methods except for ours

Results


FPS
Ours
Motion planning
(137K Tri., 50K samples)
Round-robin
Work stealing
2
1.5
1
0.5
0
1C
2C
2C+1G
2C+2G
2C+3G
2C+4G
Use different GPUs


FPS
Global illumination
(436K Tri., 80M rays)
Ours
Work stealing
Round-robin
0.08
0.06
0.04
0.02
0
1C
2C
2C+1G
2C+2G
2C+3G
2C+4G
Use different GPUs

Outline

- Motivation
- Our approach
  - Optimization-based scheduling
- Results
- **Conclusion**

Conclusion

- **Presented a novel scheduling algorithm**
  - Designed the expected running time model
  - Formulated the scheduling problem as an optimization problem
  - Proposed a novel iterative optimization solver
- **Efficiently utilized heterogeneous computing resources**
  - Achieved high scalability with additional computing resources
  - Applied to various proximity queries

Future Work

- **Extend to other general applications that have more variety of jobs**
- **Improve scheduling algorithms further**
  - – Minimize overhead and robustly handle local minimum issues
  - – Design multi-resolution scheduling for large-scale heterogeneous computing systems

References

- **[Kim 2009]** HPCCD: Hybrid Parallel Continuous Collision Detection using CPUs and GPUs, Kim et al., Computer Graphics Forum (Pacific Graphics) 2009
- **[Lee 2010]** Simple and Parallel Proximity Algorithms for General Polygonal Models, Youngeun Lee et al, Journal of Computer Animation and Virtual Worlds, 2010
- **[Govindaraju 2005]** CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware, EG. Workshop on Graphics Hardware, 2003
- **[Govindaraju 2005*]** Collision detection between deformable models using chromatic decomposition," ACM Trans. on Graphics, 2005
- **[Lauterbach 2010]** gProximity: Hierarchical GPU-based Operations for Collision and Distance Queries, C Lauterbach et al., EG 2010
- **[Tang 2009]** Multi-core collision detection between deformable models, M. Tang et al., in SIAM/ACM Joint Conf. on Geometric and Solid & Physical Modeling, 2009
- **[Wald 2007]** On fast construction of sah-based bounding volume hierarchies, I. Wald, IEEE Symposium on Interactive Ray Tracing, 2007.
- **[Ize 2007]** Asynchronous BVH construction for ray tracing dynamic scene on parallel multi-core architectures, T. Ize et al., Eurographics Symposium on Parallel Graphics and Visualization, 2007.
- **[Baciu 2002]** Image-based techniques in a hybrid collision detector, G. Baciu and S. Wong, IEEE TVCG, 2002.
- **[Lawler 2002]** A voxel-based parallel collision detection algorithm, O. S. Lawlor and V. K. Laxmikant, Super-computing, 2002.
- **[Sud 2006]** Fast Proximity Computation among Deformable Models using Discrete Voronoi Diagrams, A. Sud et al., ACM SIGGRAPH, 2006.
- **[Vassilev 2001]** Fast cloth animation on walking avatars, T. Vassilev et al., Computer Graphics Forum (Eurographics), 2001.

Thanks

Any questions?

(bluekdct@gmail.com)

Project homepage:
http://sglab.kaist.ac.kr/hybrid_parallel
(This presentation slides are available at the homepage)

* This work was published at IEEE TVCG
and selected as the **Spotlight paper** for the Sept. 2013 issue