Proximity Computations on Heterogeneous Computing Systems

GTC 2013

Duksu Kim

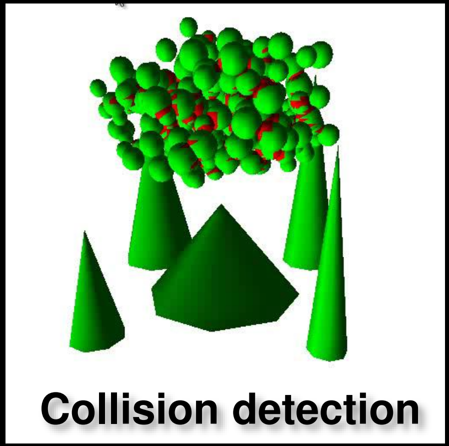
Ph. D. candidate, Scalable Graphics Lab.,

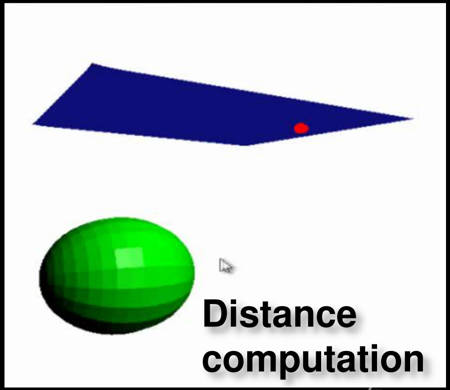
Korea Advanced Institute of Science and Technology (KAIST)

Proximity Queries (PQs)

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

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

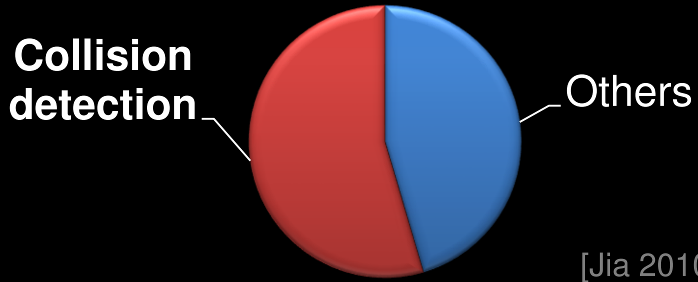




Proximity Queries in App.

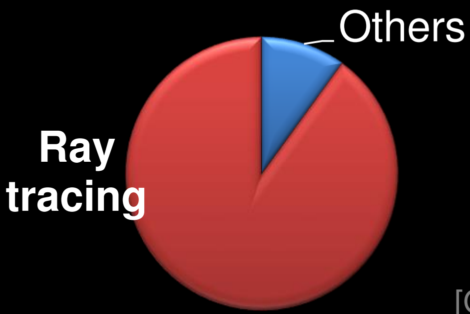
from Bochang's paper

Motion planning



[Jia 2010] [Liangjun 2008]

Realistic redering



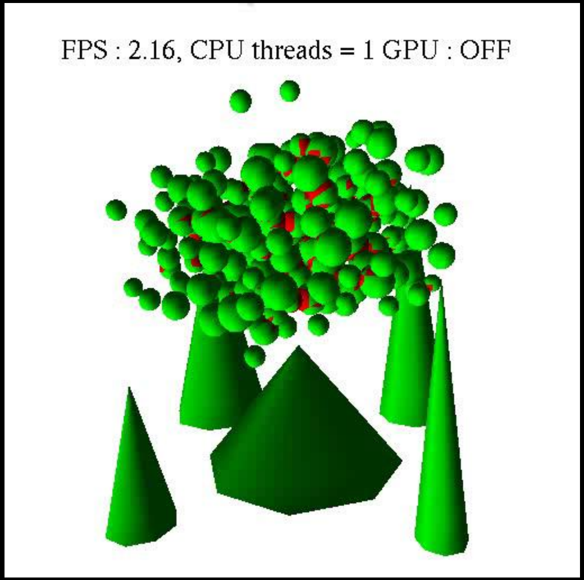
[Our in-house render]

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

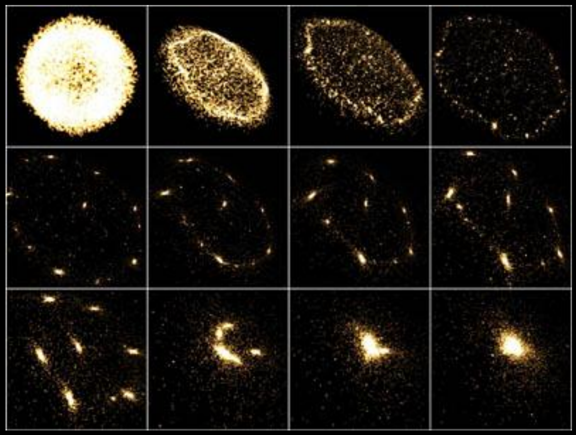
N-body benchmark
consisting of 34K triangles

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 requires interactive/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)

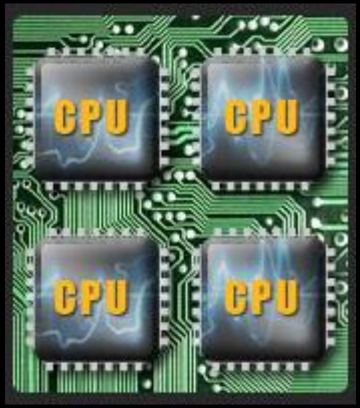


Image from NVIDIA

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

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*]

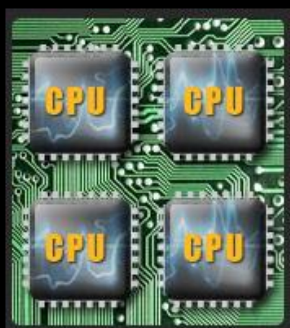
Our Goals & Approaches

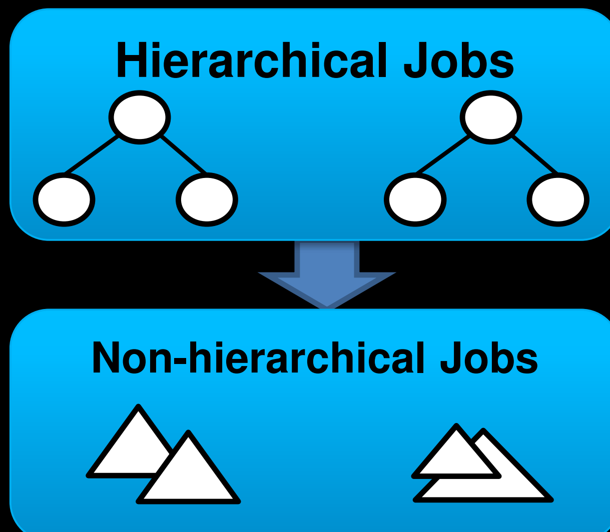
- **Achieve real-time performance**
  - In various proximity queries for large-scale models

- **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:

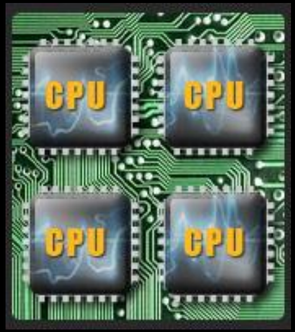
HPCCD:
Hybrid Parallel Continuous Collision Detection





Previous Work:

HPCCD:
Hybrid Parallel Continuous Collision Detection



- Branch prediction
- Cache

Hierarchical Jobs

Random accesses

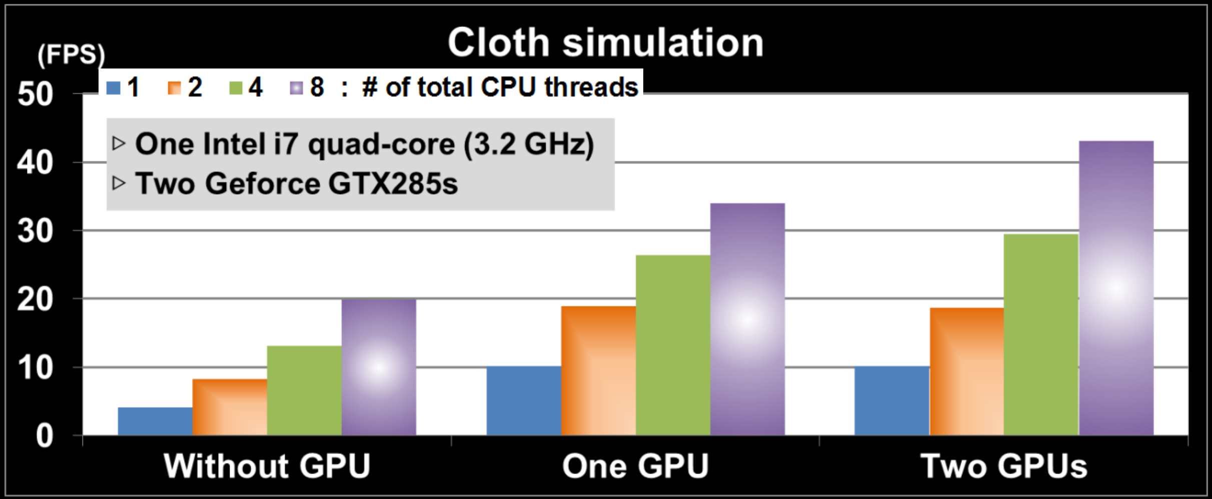
Non-hierarchical Jobs

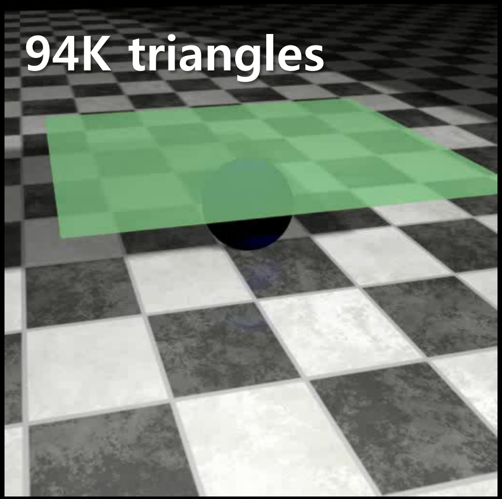
Solving cubic equations

- Massive parallelism

Previous Work:

Results



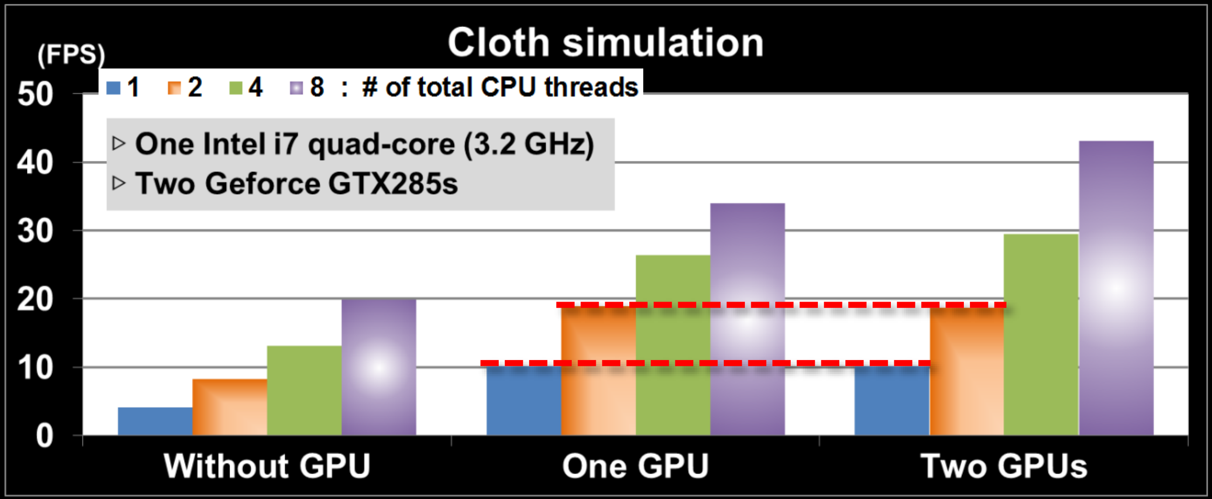


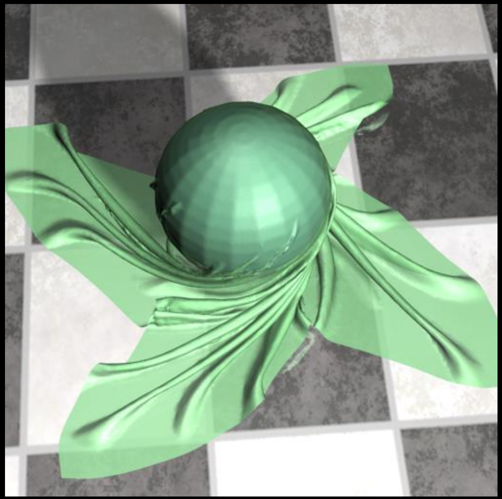
- **Manually specify distribution rules depending on knowledge on the application**

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

Previous Work:

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

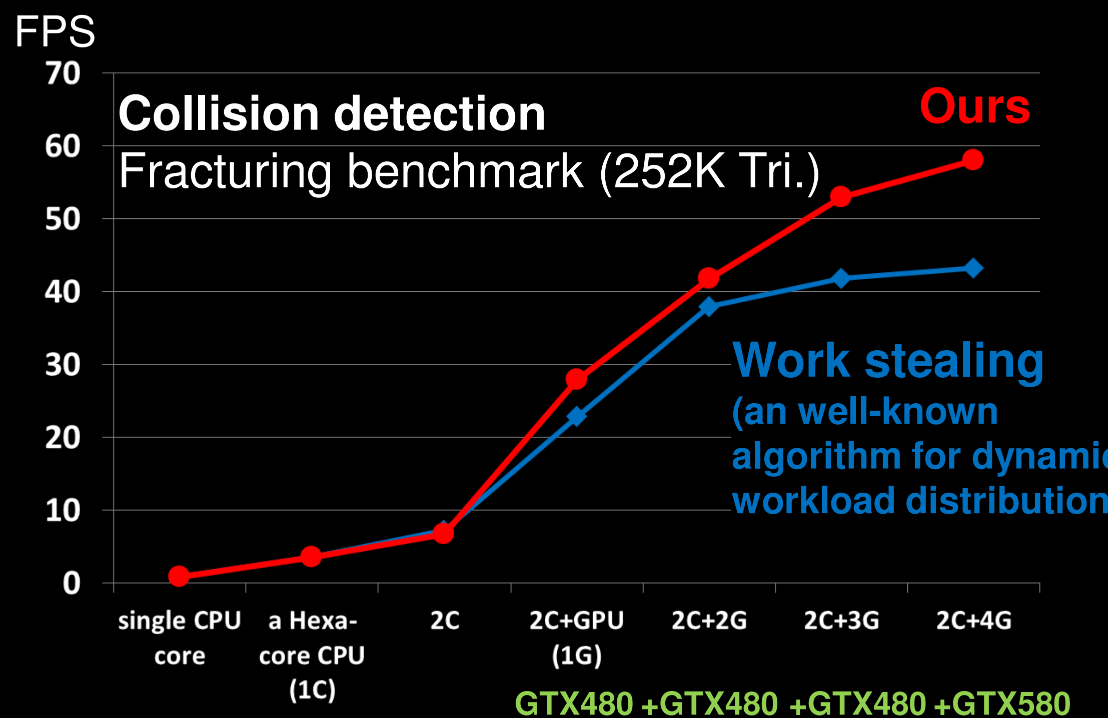
Current Work

- Previous work: Manual scheduling

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

Current Work – Results

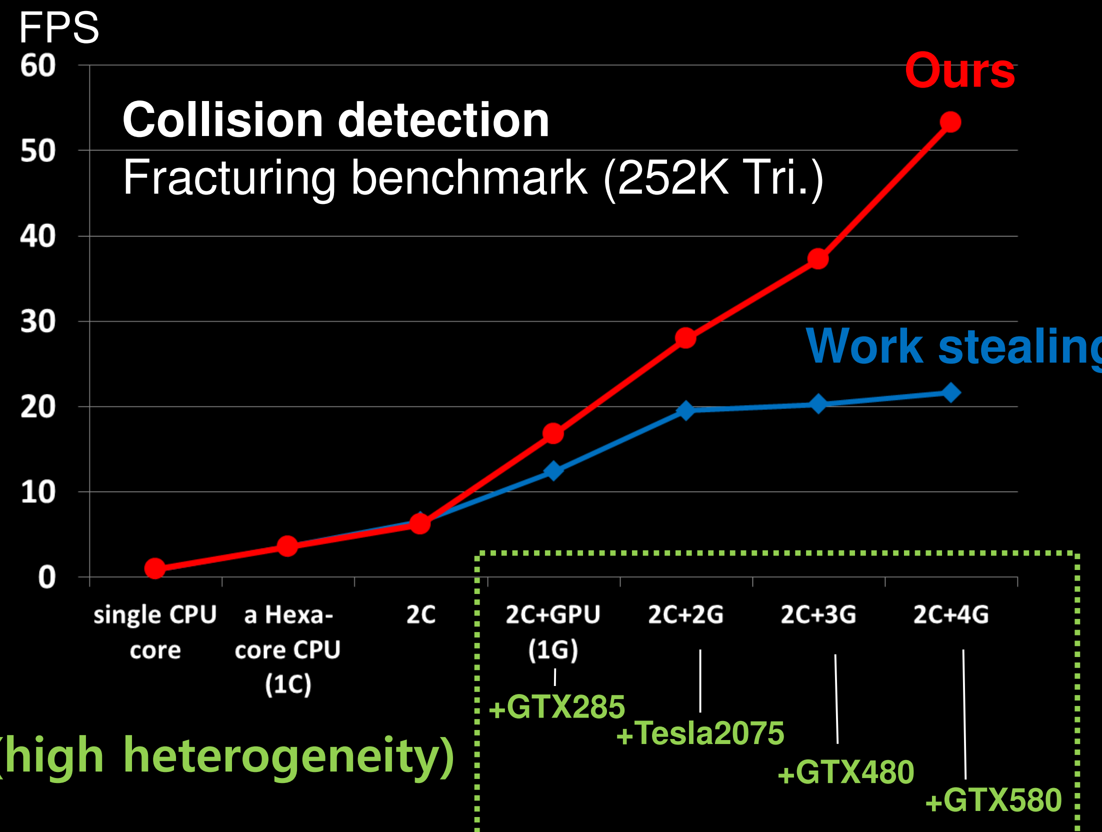
(With our optimization-based scheduling)



Use same GPUs (low heterogeneity)

Current Work – Results

(With our optimization-based scheduling)

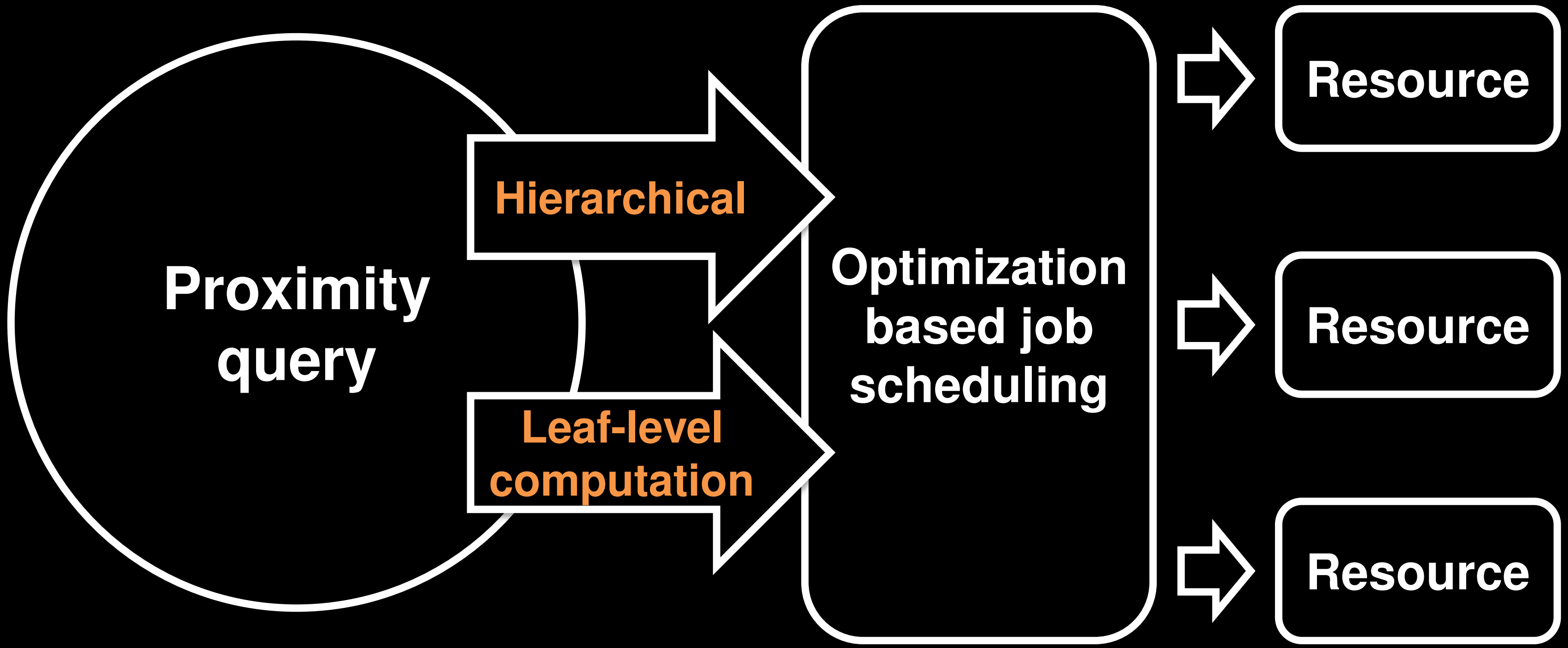


Use different GPUs (high heterogeneity)

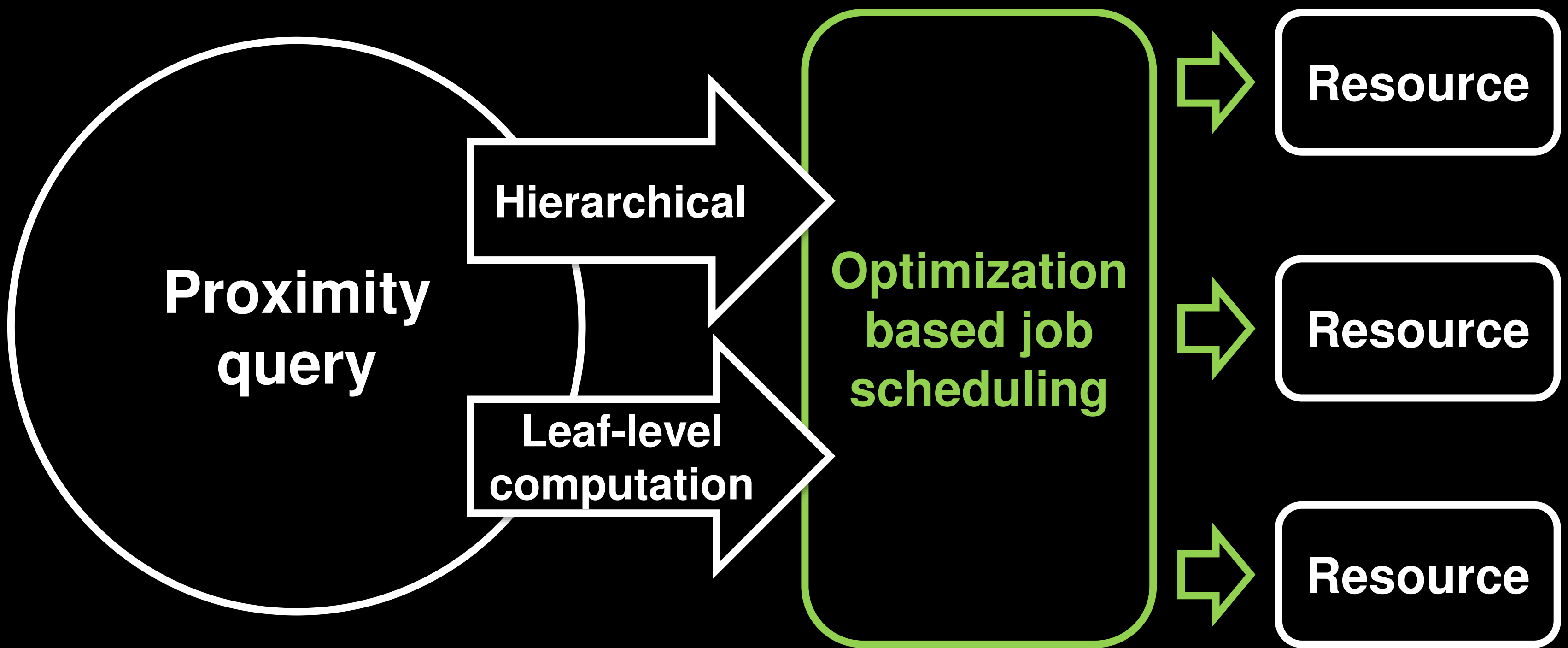
Outline

- 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

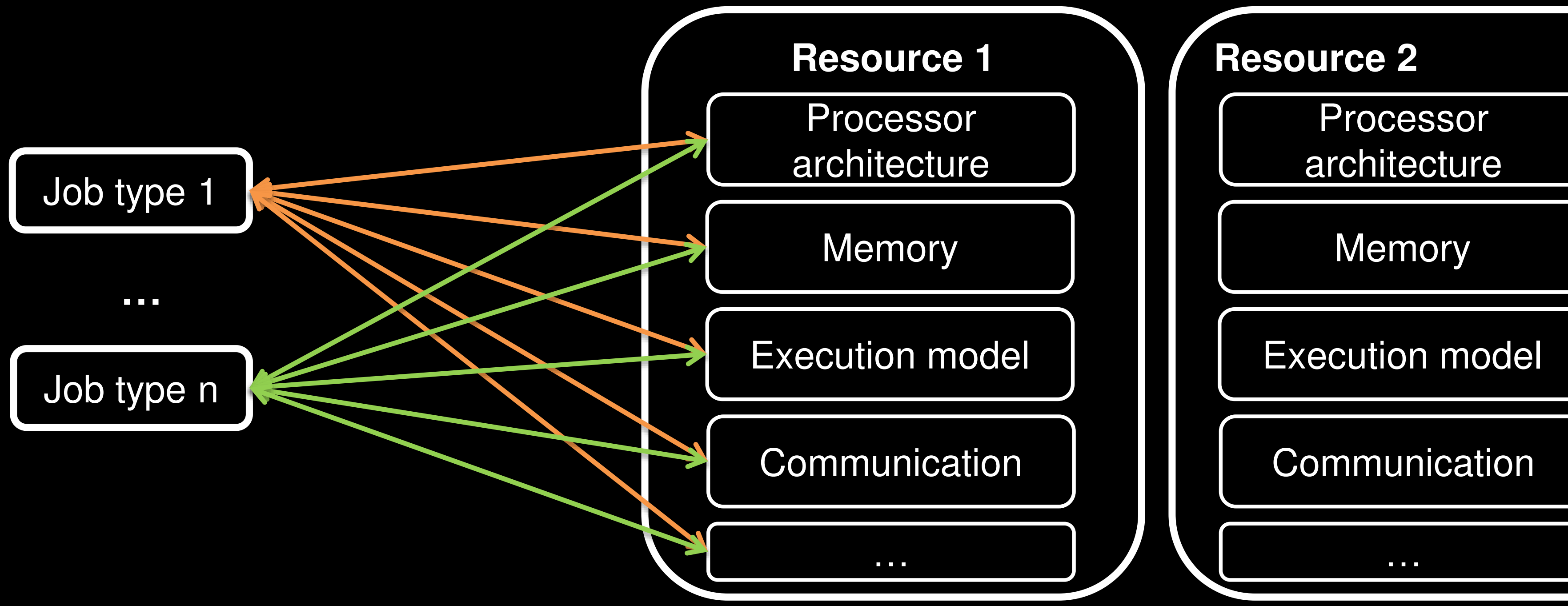
Optimization-based Scheduling

Expected running time model → Optimization formulation → Iterative LP solver

- **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

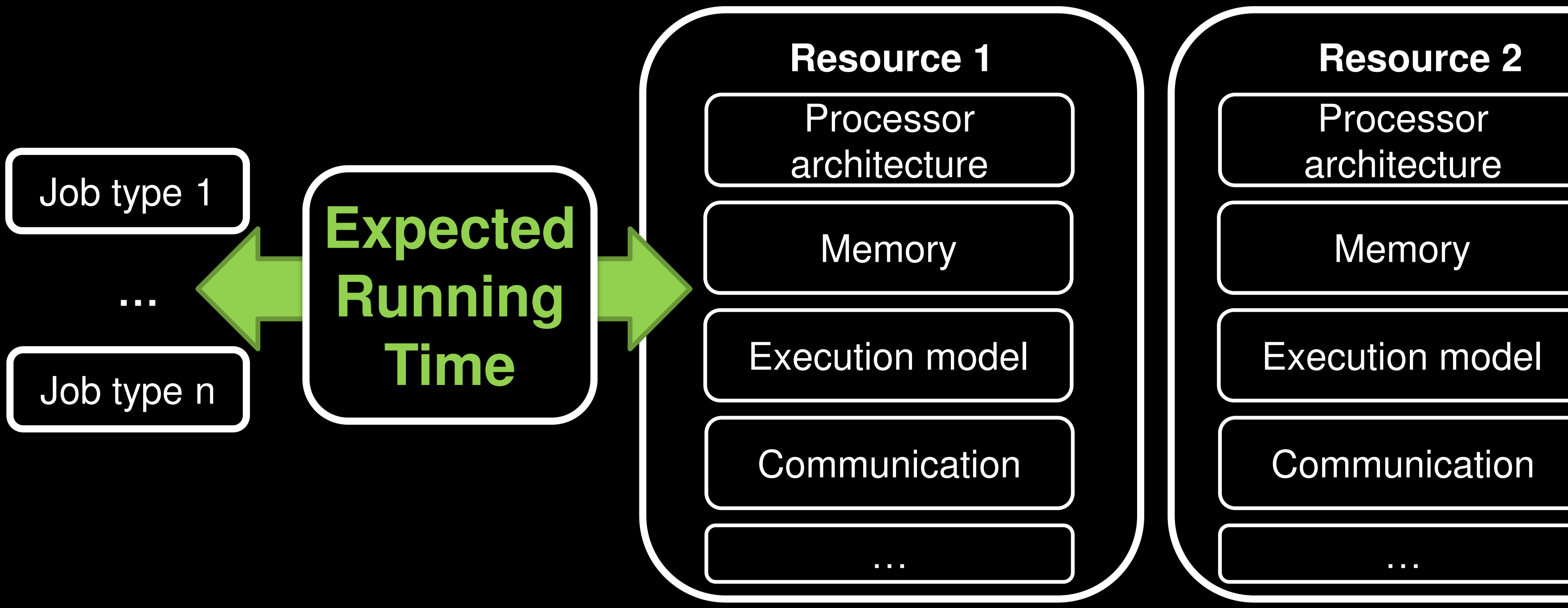
Performance Model

- **Complex relationship between jobs and resources**

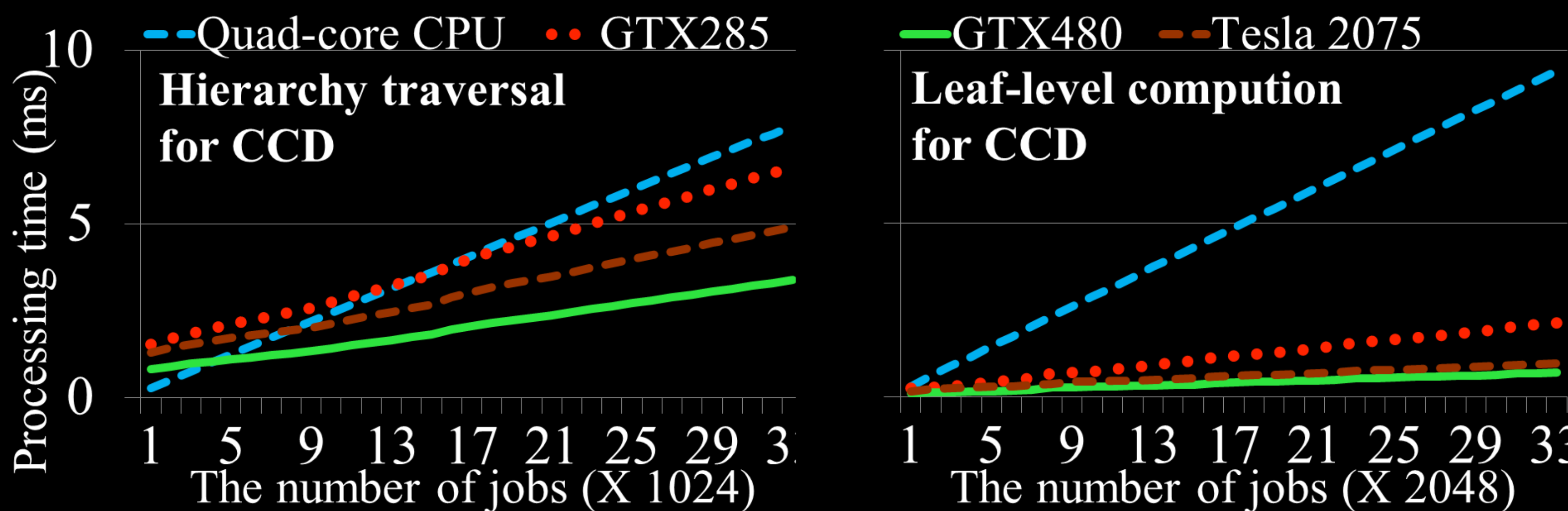


Performance Model

- **Abstract the complex relationship as an expected running time model**

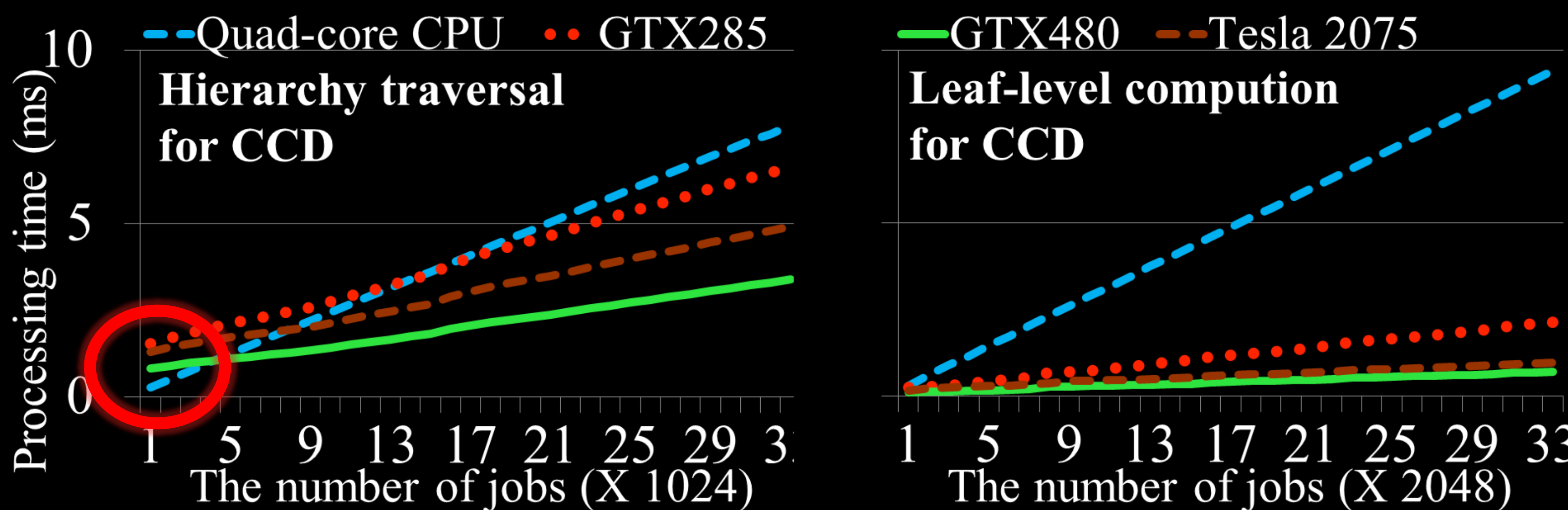


Performance Model



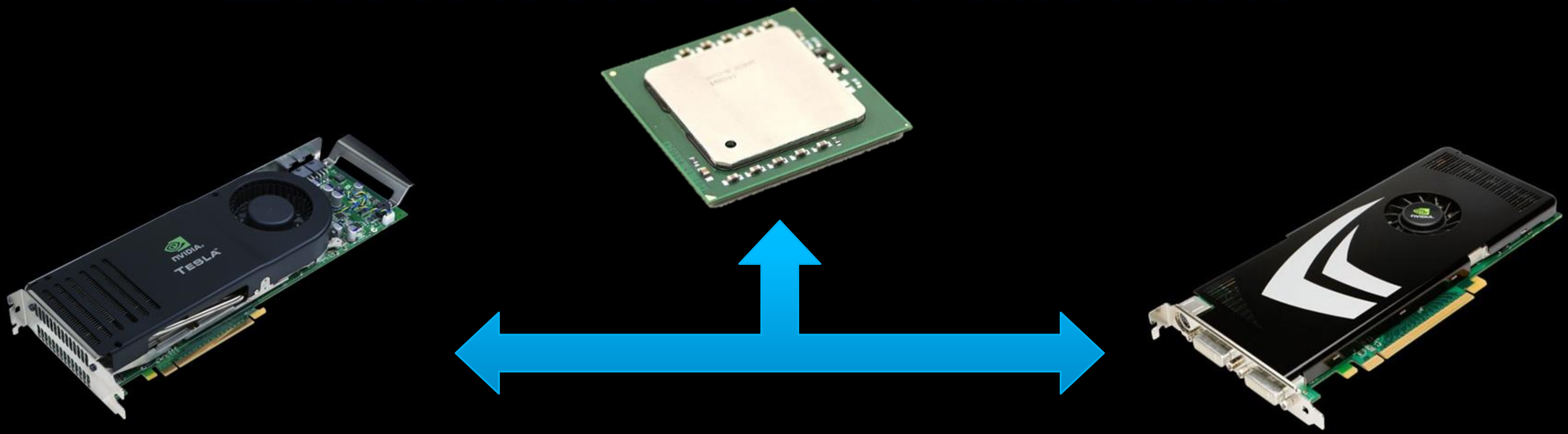
- **Running time is linearly increased as the number of jobs is increased**

Performance Model



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

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

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

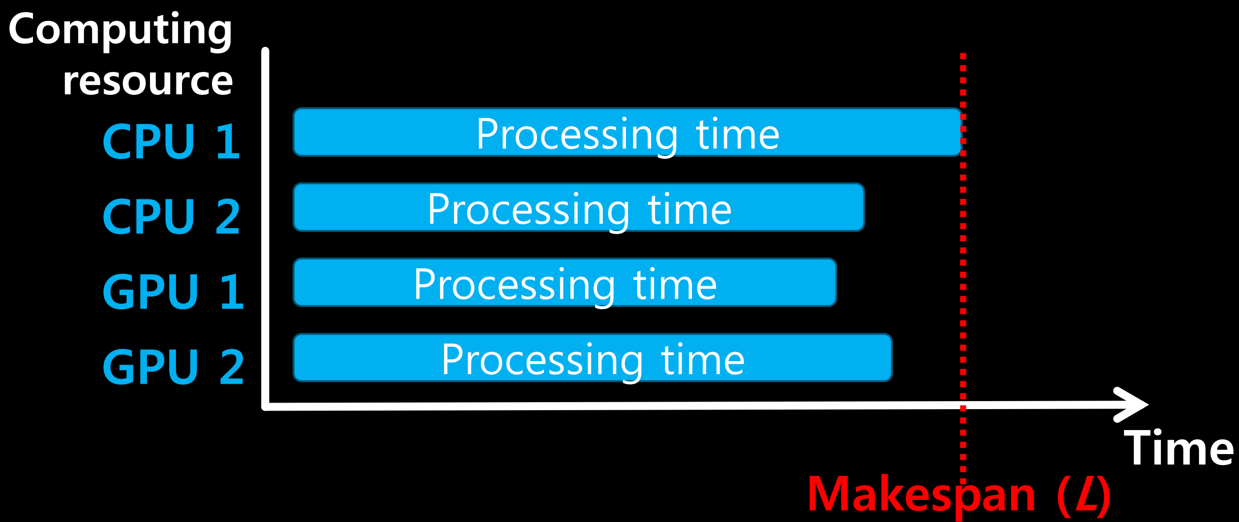
$$T(k \rightarrow i, j, n_{ij}) = \begin{cases} 0, & \text{if } n_{ij} \text{ is } 0 \\ T_{setup}(i,j) + T_{proc}(i,j) \times n_{ij} \\ \quad + T_{trans}(k \rightarrow i, j) \times n_{ij}, & \text{otherwise.} \end{cases}$$

- **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 makespan problem**

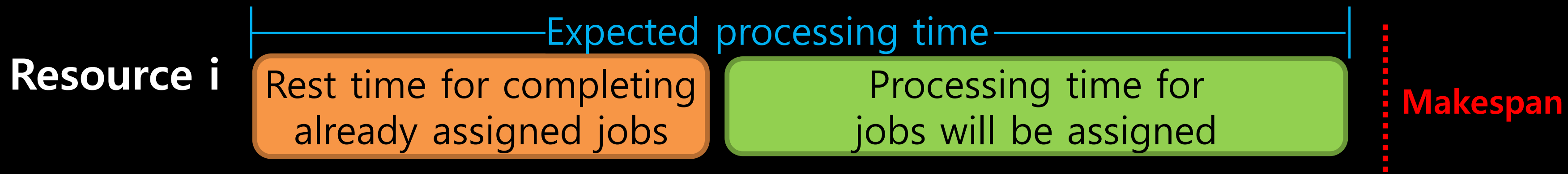
$$Minimize\ L,$$



Optimization Formulation

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

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



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 is no missing or duplicated jobs

Optimization Formulation

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

$$Minimize\ L,$$

Job distribution

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

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

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

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

② There is no missing or duplicated jobs

③ Each job is atomic

Optimization-based Scheduling

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

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

Optimization-based Scheduling

Expected running time model → Optimization formulation → Iterative LP solver

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

NP-hard Problem!

- **High computational cost**
  - Jobs are dynamically generated at runtime
  - Optimization process takes long time to apply to 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 iterative solve 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 technical report for the details (http://sglab.kaist.ac.kr/Hybrid_parallel)

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

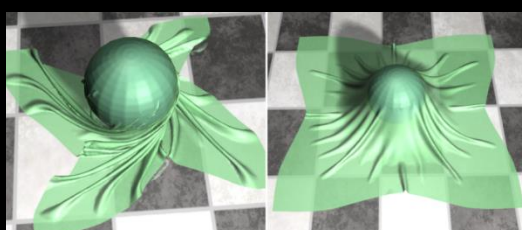
Positive floating-point numbers

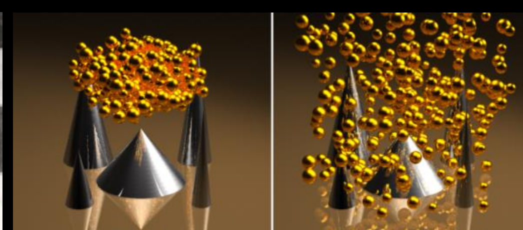
Outline

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

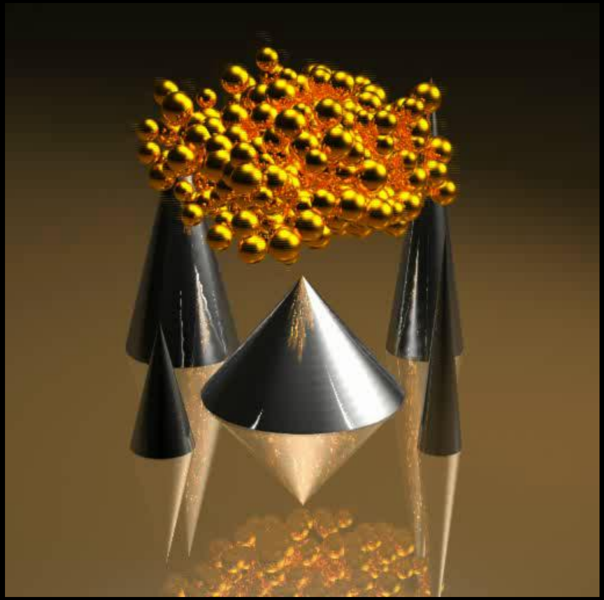
Results

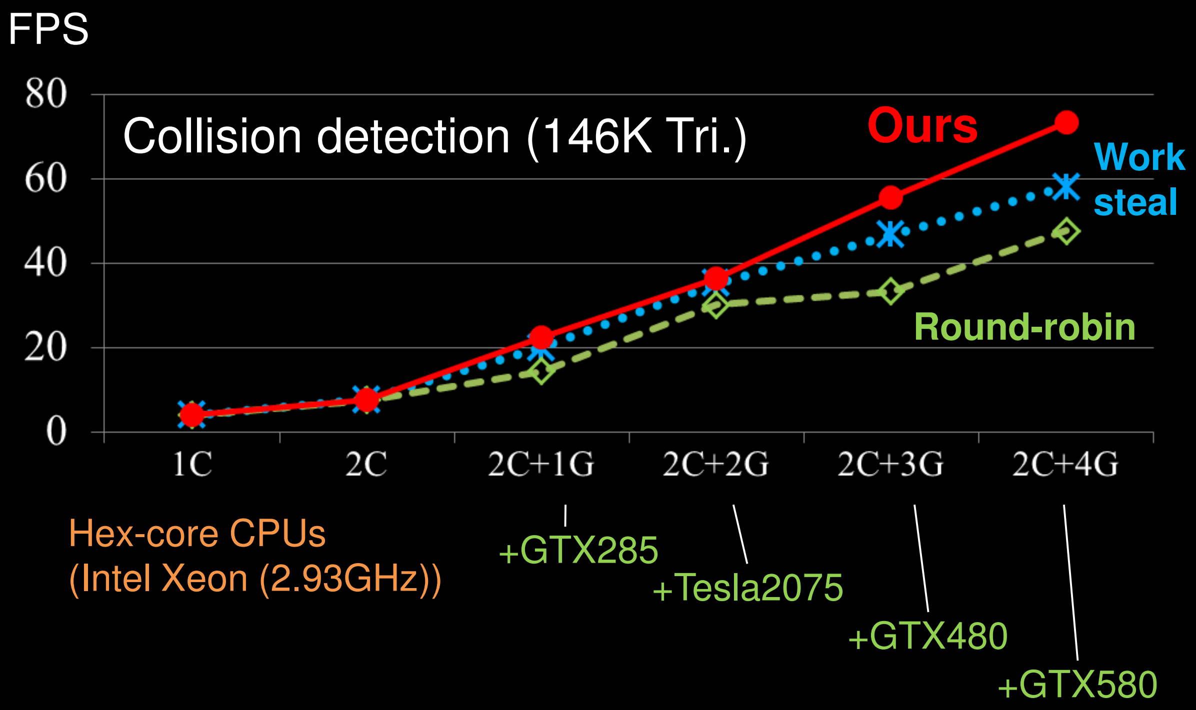
- **Applied to various application**
  - Collision detection
  - Motion planning
  - Global illumination





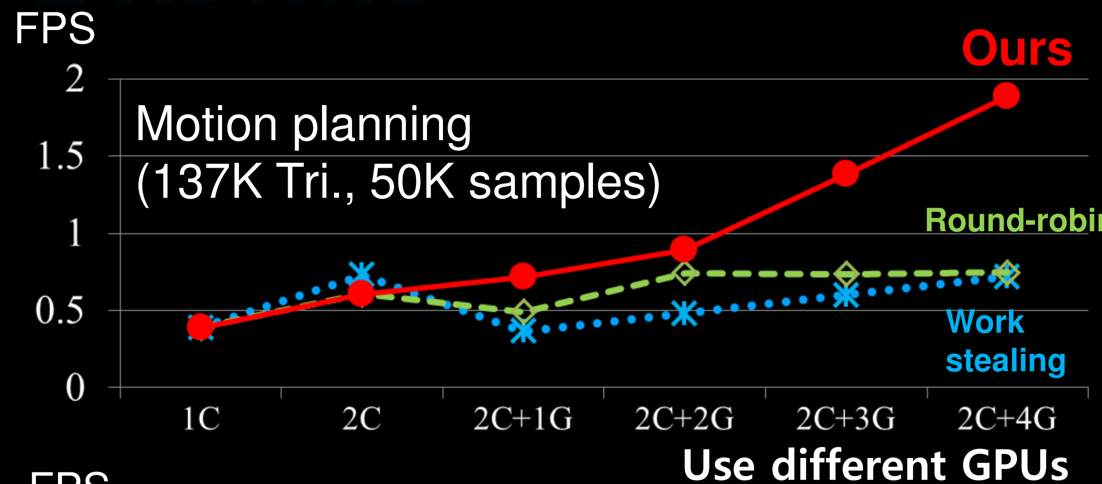
Results

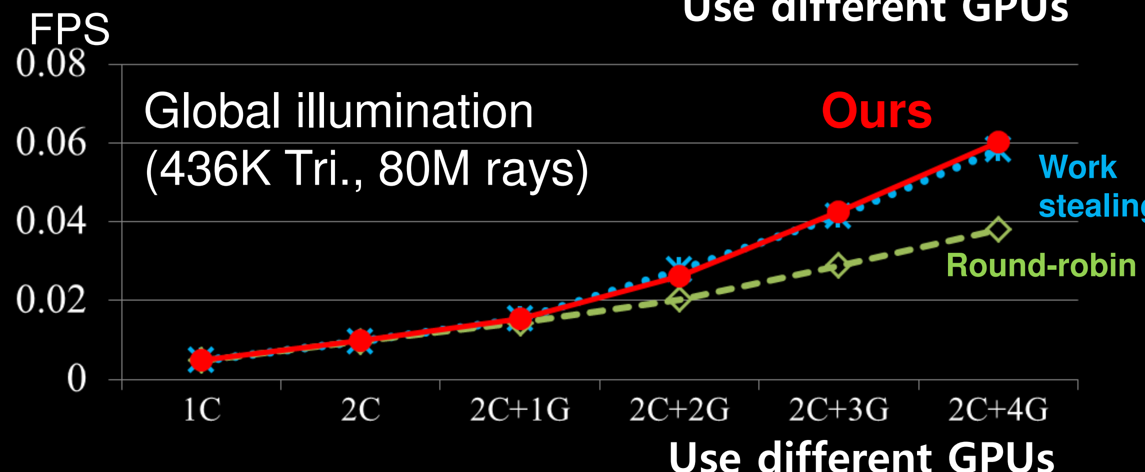




Use different GPUs

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

Outline

- 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

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