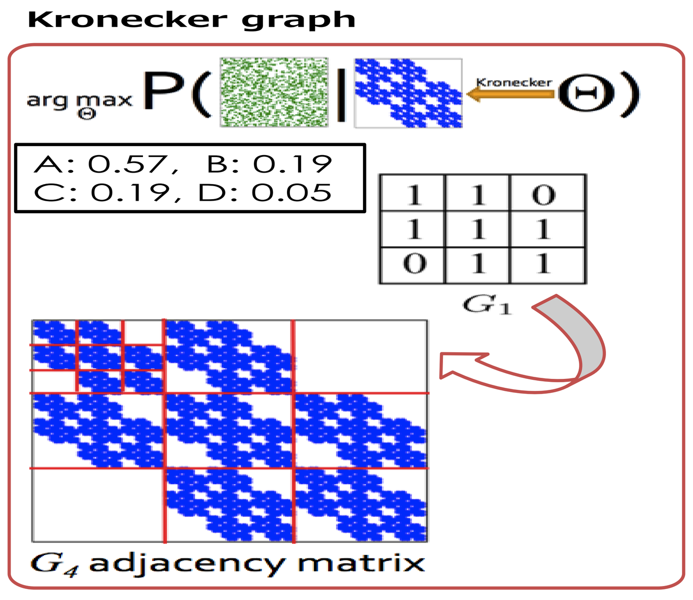
TOKYO TECH
Pursuing Excellence

Data-Intensive Supercomputing

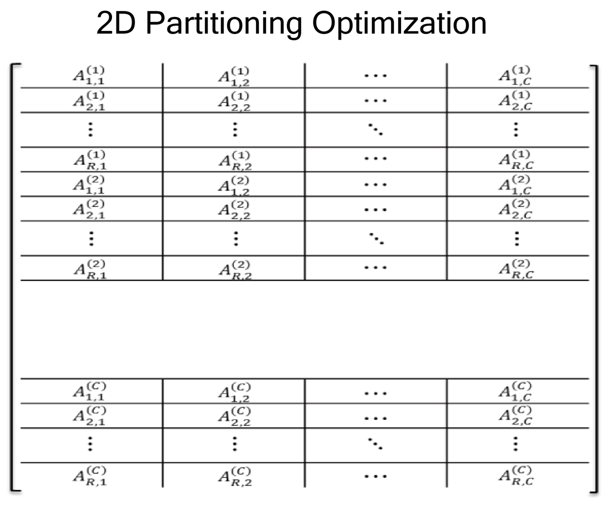
Graph500 Challenges

Graph500 with 4000 GPUs

- Graph500 is a benchmark that ranks supercomputers by executing a large-scale graph search problem.
- The benchmark is ranked by so-called TEPS (Traversed Edges Per Second) that measures the number of edges to be traversed per second by searching all the reachable vertices from one arbitrary vertex with each team's optimized BFS (Breadth-First Search) algorithm.

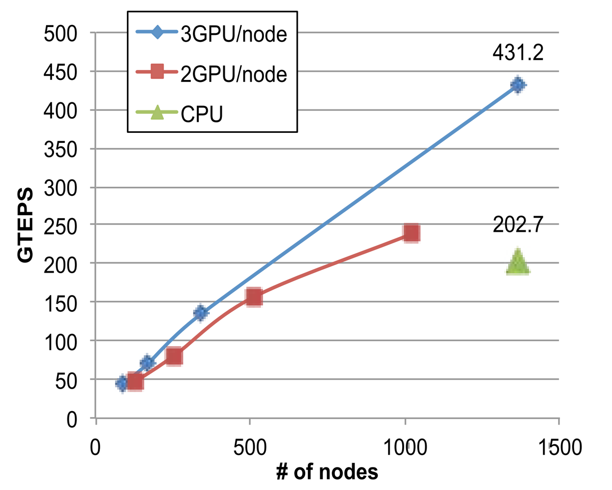


2D Partitioning Optimization



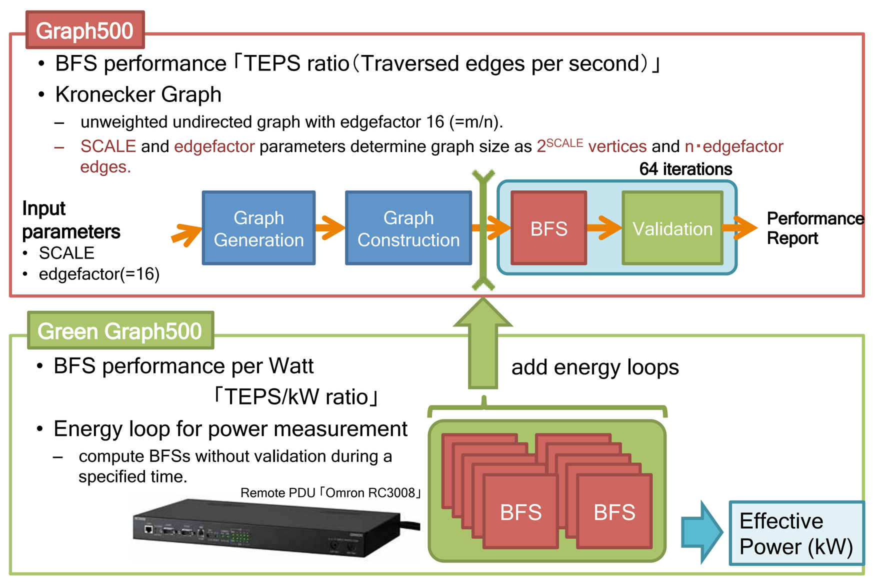
- We propose an optimized method based on 2D based partitioning and other various optimization methods such as communication compression and vertex sorting.
- We developed CPU implementation and GPU implementation.
- Our optimized GPU implementation can solve BFS (Breadth First Search) of large-scale graph with 2^{35} (34.4 billion) vertices and 2^{39} (550 billion) edges for 1.275 seconds with 1366 nodes and 4096 GPUs on TSUBAME 2.0.
- This record corresponds to 431 GTEPS.

Performance Comparison with CPU and GPU implementations



Green Graph500

- Green Graph500 is a new benchmark that complements Graph500 with an energy metric.
- The benchmark is ranked by TEPS/kW that measures BFS computation time in Graph500 without validation during a specified time.



SCALE 26 (#vertices: 67,108,864 ≈2^{26} 、#directed-edges: 2,147,475,066 ≈2^{31})

Machine (#threads)	TEPS ratio	CPU time* (s)	Effective Power (W)	Apparent Power (kVAh)	CO2 (kg)	TEPS/kW
WestmreEX (80)	10.066 G	207.21	1000.6	0.06	0.02	10.060 G
SandyBridgeEP (32)	5.607 G	370.34	362.0	0.04	0.01	15.489 G

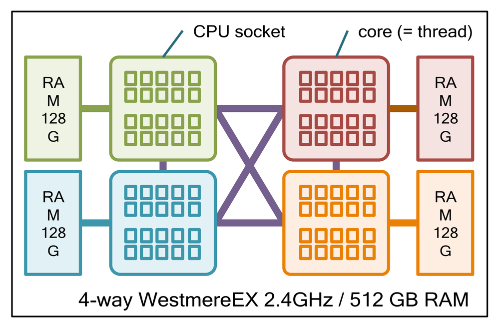
SCALE 21 (#vertices: 2,097,152 ≈2^{21} 、#directed-edges: 67,105,930 ≈ 2^{26}) ≈ TIGER/Line US road-network

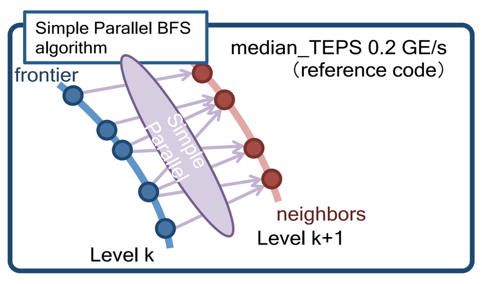
Machine (#threads)	TEPS ratio	CPU time* (s)	Effective Power (W)	Apparent Power (kVAh)	CO2 (kg)	TEPS/kW
WestmreEX (80)	3.967 G	6.06	854.1	0.00	0.00	4.645 G
SandyBridgeEP (32)	5.293 G	6.23	310.3	0.00	0.00	17.058 G
Atom2 (2)	0.163 G	925.16	94.6	0.03	0.01	1.723 G
MacbookAir (4) Intel Core i5-2557M 1.70GHz	0.782 G	375.93	20.3	0.00	0.00	38.133 G

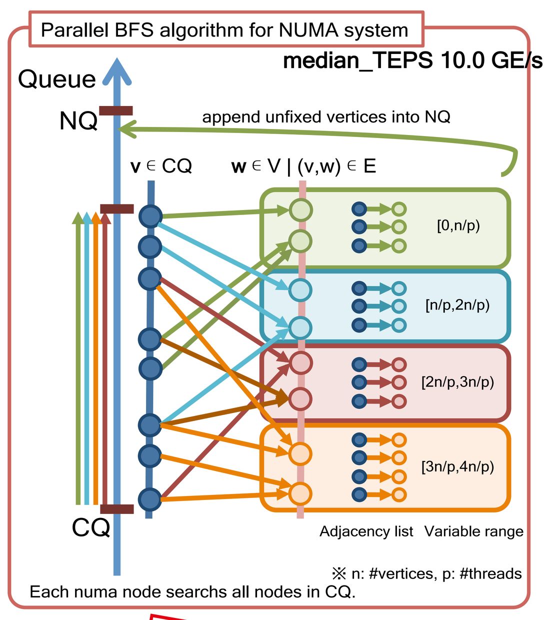
* CPU time contains generation, construction, BFSs, and validations.

- Our early study reveals that HPC processors such as Intel Westmere EX and SandyBridge EP are energy efficient than low power processors such as Intel Atom.

- We have implemented a high-efficient parallel BFS algorithm for NUMA systems, which pins CPU cores on NUMA-local RAM using CPU affinity.





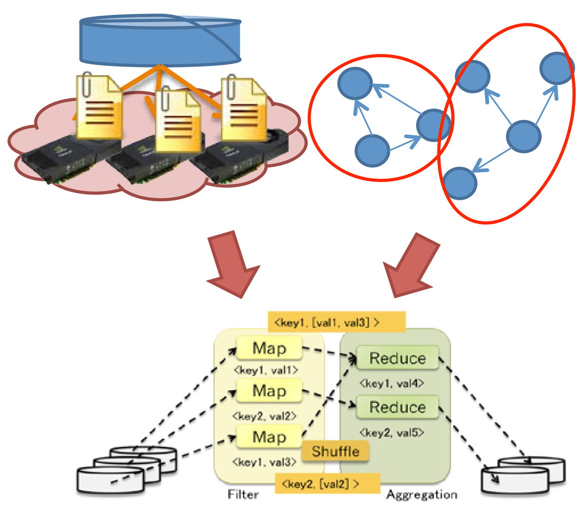


Cooperative work with Katsuki Fujisawa, Chuo Univ.

GPU MapReduce

Multi-GPU Implementation of GIM-V

- GIM-V (Generalized Iterative Matrix-Vector multiplication)
  - MapReduce-based graph algorithm (PageRank, CC, RWR)
  - General expression of matrix-vector multiplication with iterative operations



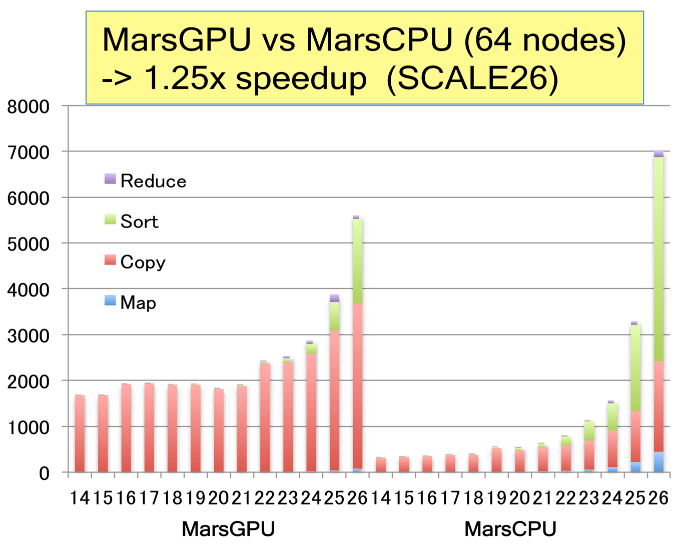
Problem

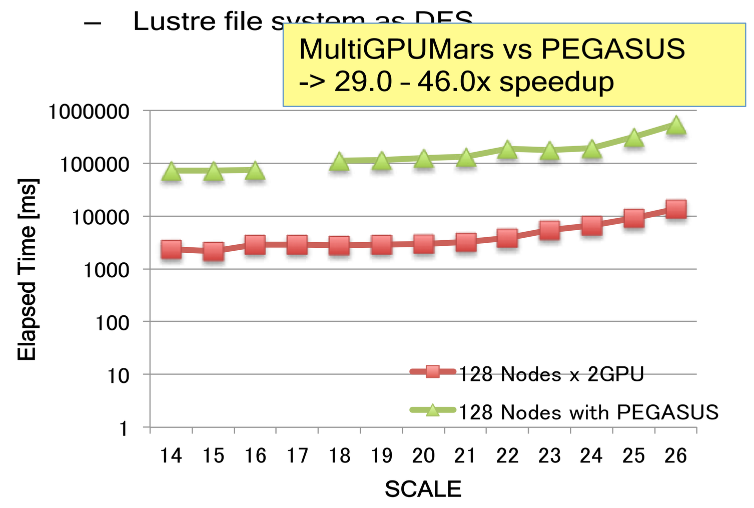
- Scalability and data management optimization on a multi-GPU environment

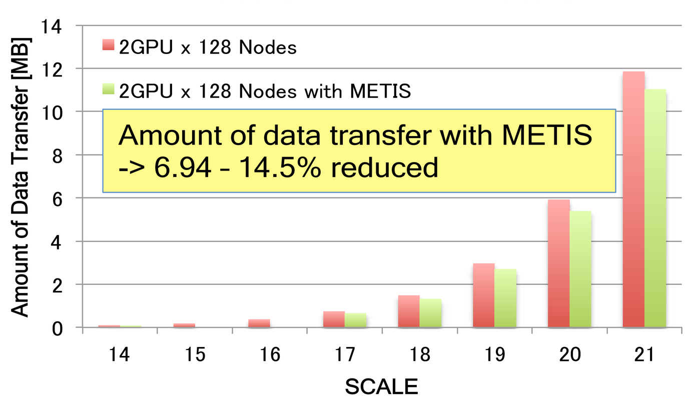
Solution

- Multi-GPU-based implementation for handling extremely large-scale graph
  - We extend the existing single-GPU-based MapReduce library (Mars) for GPU-based heterogeneous supercomputers
- Data transfer reduction in communication between GPU devices by using a graph partition method (METIS)

Experiments
- TSUBAME 2.0
  - Up to128 nodes (3072 hyper-threaded CPU cores, 384 GPUs)
- Mars
  - *MarsGPU*: 3 GPUs / node
  - *MarsCPU*: 24 threads / node
- PEGASUS
  - Hadoop 0.21.0
  - Lustre file system as DFS





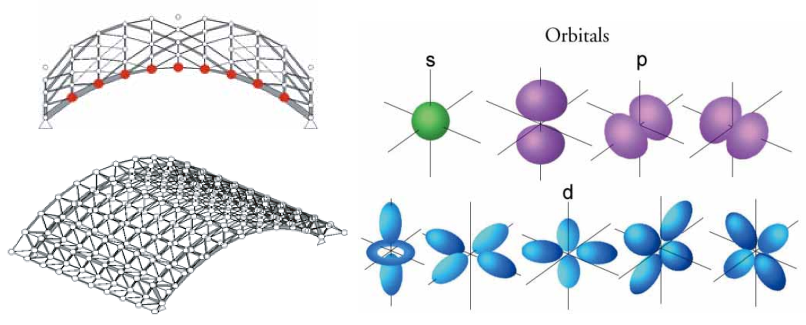


Mathematical Optimization

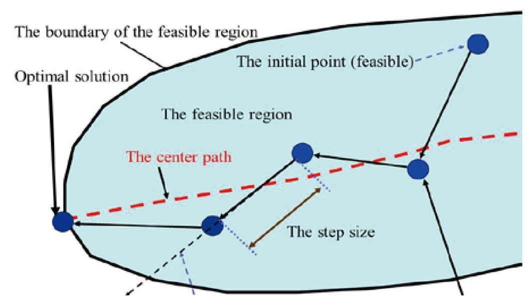
SC12 Tech Paper Thu 2:30pm Room 355-EF

Solving SDP problems with > 1M constraints

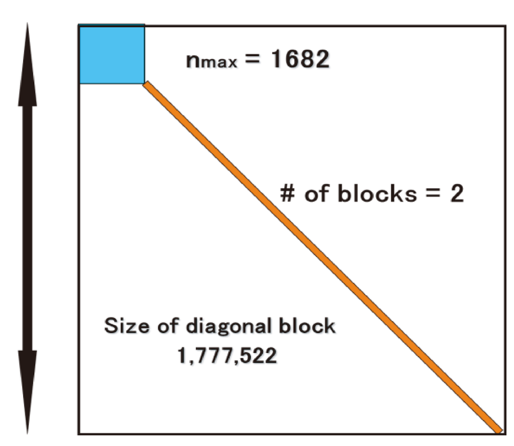
- SDP (semidefinite programming) is one of mathematical programming methods with many applications:
  - structural optimization, combinatorial optimization, quantum chemistry, etc.

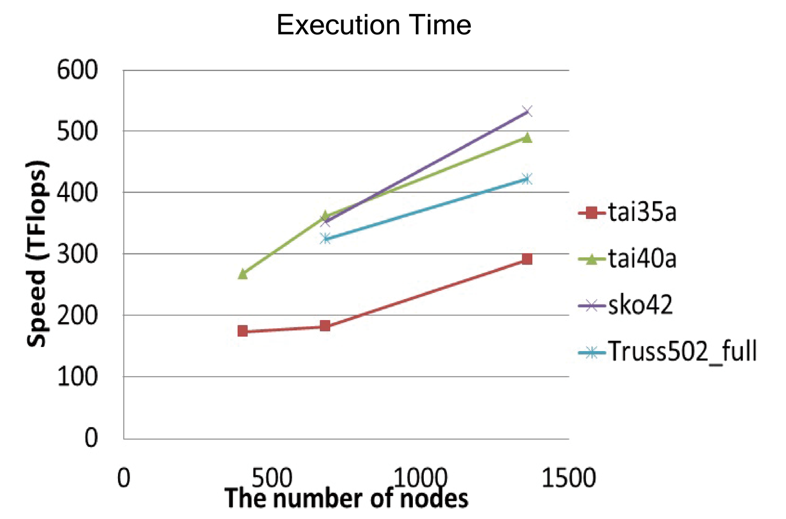


- SDPARA is a parallel implementation of the primal-dual interior-point method for SDP
- SDPARA can solve the largest SDP problem that has over 1.48 million constraints (sko42a problem), and create a new world record in 2012
  - Using 1360 nodes, 2720 CPUs, 4080 GPUs
  - Computation time: 2700 sec./iterations (About 30 iterations are required)
  - Performance of CHOLESKY: 2045 sec./iteration, 533TFLOPS

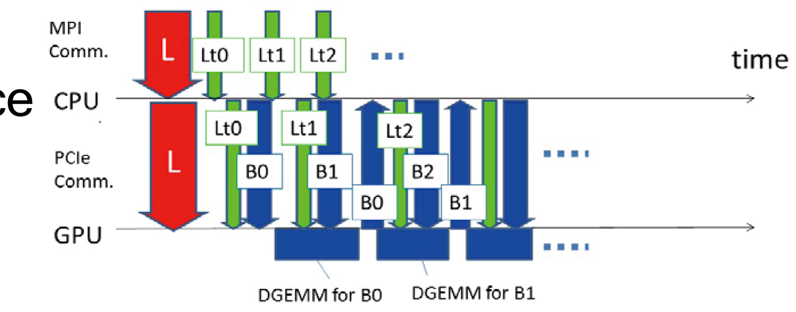


n = 1,779,204
(matrix size)
m = 1,484,406
(# of constraints)
nnz = 23,476,376
(# of nonzero elements)





- Accelerate CHOLESKY by using massively parallel GPUs
  - In order to achieve scalable performance with thousands of GPUs
  - Utilize high performance BLAS kernel coupled with optimization techniques to overlap computation, PCI-E communication and MPI communication.



Cooperative work with Katsuki Fujisawa, Chuo Univ.

http://www.gsic.titech.ac.jp/sc12