PSHybrid Domain Parallel Algorithm for 3D Kirchhoff Prestack Depth Migration*

Changhai Zhao[1,2], Jianlei Zhang[2], Guoan Luo[3], Quanshun Cui[1,2], and Chengxiang Wang[1]

Search and Discovery Article #42040 (2017)**
Posted March 20, 2017

*Adapted from poster presentation given at AAPG International Conference & Exhibition, Barcelona Spain, April 3-6, 2016

[1]BGP (zhao.ch@cnpc.com.cn)
[2]CNPC
[3]University of Utah

Abstract

The size of seismic data from a survey area has exceeded 100 TB, and will increase to more than 1 PB in the near future. The size of PC cluster used for seismic data processing has been 256~1000 nodes and GPU cluster has been used to speed up the parallel computation. To support increasing survey sizes and processing complexity, we propose a practical approach of implementing the large-scale parallel processing of 3D Kirchhoff Prestack Depth Migration (KPSDM) on heterogeneous cluster. The parallel algorithm is called "hybrid domain parallel", which is based on three-level decomposition including offset, imaging space, and seismic data. The algorithm eliminates the dependencies among tasks. We design a "dynamic and asynchronous" task allocation policy to make the I/O of waiting tasks and the computation of current tasks asynchronous execution and achieve load balancing on heterogeneous computing system.

Because KPSDM, as part of its execution, usually requires repeated access to huge seismic data and a large travel time table data base, the scalability is always limited by the shared storage maximum throughput. To solve the scalability problem, we build a distributed cache system using local storage of all hosts that a KPSDM job spans. The distributed cache system stores the seismic data and travel time table data corresponding to each task on the local storage. According to the load of each node, it can select the spare node to provide a very high aggregate data bandwidth and supply seismic data and travel time table data to a task timely. In order to make the KPSDM more robust, MPI model is substituted for the Geophysical Parallel Programming (GPP) model which can support the task migration in the execution of application program. Furthermore, the "task backup" strategy from MapReduce frame is used for slow nodes to avoid the delay of job running time. The KPSDM implementation can obtain close to linear speedup when it processes real word data on a 256-node cluster.

References Cited

Addair, T.G., D.A. Dodge, W.R. Walter, and S.D. Ruppert, 2014, Large-scale seismic signal analysis with Hadoop: Computers & Geosciences, v. 66, p. 145-154.

Chang, H., J.P. VanDyke, M. Solano, G.A. McMechan, and D. Epili, 1998, 3-D prestack Kirchhoff depth migration: From prototype to production in a massively parallel processor environment: Geophysics, v. 63, p. 546-556.

Huazhong, Wang, Liu Shaoyong, Kong Xiangning, Cai Jiexiong, and Fang Wuba, 2012, 3D Kirchhoff PSDM for large-scale seismic data and its parallel implementation strategy: OGP, v. 47, p. 404-410.

Rastogi, R., A. Srivastava, K. Khonde, K.M. Sirasala, A. Londhe, and H. Chavhan, 2015, An efficient parallel algorithm: Poststack and prestack Kirchhoff 3D depth migration using flexi-depth iterations: Computers & Geosciences, v. 80, p. 1-8.

Panetta, J., P.R.P. de Souza Filho, C.A. Da Cunha Filho, F.M.R. Da Motta, S.S. Pinheiro, I. Pedrosa, 2007, Computational characteristics of production seismic migration and its performance on novel processor architectures: 19th International Symposium on Computer Architecture and High Performance Computing, p. 11-18.

Panetta, J., T. Teixeira, P.R.P. de Souza Filho, C.A. Da Cunha Filho, D. Sotelo, and F.M. Roxo Da Motta, 2011, Accelerating time and depth seismic migration by CPU and GPU cooperation: International Journal of Parallel Programming, p. 1-23.

Li, J., D. Hei, and L. Yan, 2009, Partitioning algorithm of 3-D prestack parallel Kirchhoff depth migration for imaging spaces: Eighth International Conference on Grid and Cooperative Computing, p. 276-280.

Cunha, C.A., J. Pametta, A. Romanelli, and I. Pedrosa, 1995, Compression of traveltime tables for prestack depth migration: presented at the SEG Technical Program, Expanded Abstracts.

Alkhalifah, T., 2006, The many benefits of traveltime compression for 3D prestack Kirchhoff migration: 68th EAGE Conference & Exhibition,.

Alkhalifah, T., 2011, Efficient traveltime compression for 3D prestack Kirchh off migration: Geophysical Prospecting, v. 59, p. 1-9.

Davide Teixeira, A.Y., and Sampath Gajawada, 2013, Implementation of Kirchhoff prestack depth migration on GPU: presented at the SEG Technical Program, Expanded Abstracts.

Schroeder, B., and G.A. Gibson, 2010, A large-scale study of failures in high-performance computing systems: IEEE Transactions on Dependable and Secure Computing, v. 7, p. 337-350.

Cappello, F., A. Geist, W. Gropp, S. Kale, B. Kramer, and M. Snir, 2014, Toward exascale resilience: 2014 update: Supercomputing frontiers and innovations, v. 1.

Dean, J., and S. Ghemawat, 2008, MapReduce: simplified data processing on large clusters: Commun. ACM, v. 51, p. 107-113.

Qi, C., L. Cheng, and X. Zhen, 2014, Improving MapReduce performance using smart speculative execution strategy: IEEE Transactions on Computers, v. 63, p. 954-967.

Hybrid Domain Parallel Algorithm for 3D Kirchhoff Prestack Depth Migration

Zhao Changhai, Cui Quanshun, Zhang Jianlei, Luo Guoan, Wang Shihu, Wang Chengxiang
BGP, CNPC
zhao.ch@cnpc.com.cn

Abstract

◆ To support increasing survey sizes and processing complexity, we propose a practical approach that implementing the large-scale parallel processing of 3D Kirchhoff Prestack Depth Migration (KPSDM) on heterogeneous cluster.
◆ The parallel algorithm is called "hybrid domain parallel"，which is based on three-level decomposition including offset, imaging space, and seismic data. The algorithm eliminates the dependencies among tasks. We design a "dynamic and asynchronous" task allocation policy to achieve load balancing on heterogeneous computing system.
◆ Because KPSDM, as part of its execution, usually requires repeated access to huge seismic data and a large travel time table data base, the scalability is always limited by the shared storage maximum throughput. To solve the scalability problem, we build **a distributed cache system using local storage of all hosts** that a KPSDM job spans. It can provide a very high aggregate data bandwidth to supply seismic data and travel time table to a task timely.
◆ The KPSDM implementation can obtain close to linear speedup when it processes real word data on a 256-node cluster.

The Architecture of Heterogeneous Cluster

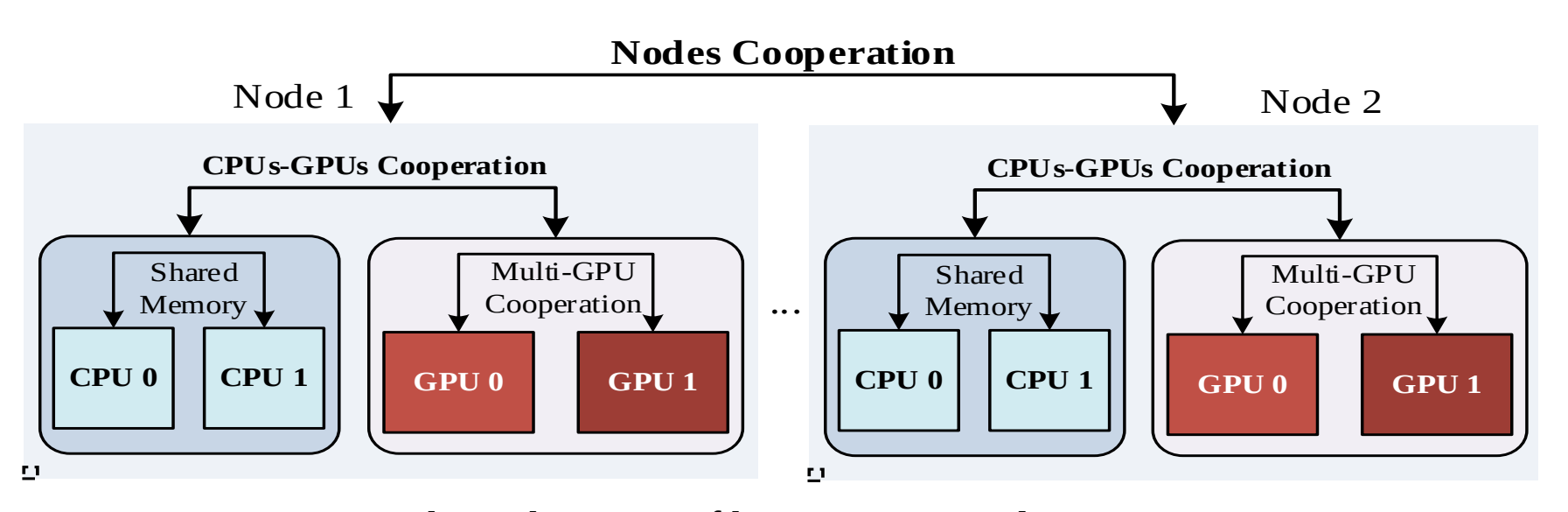

Fig 1. The architecture of heterogeneous cluster

➢ Generally, the scale of a single cluster in the processing center has reach 256 nodes. The scale of the cluster in a large processing center has reached thousands of nodes;
➢ The consolidated storage is used to store the seismic data;
The computing nodes are ordinary dual x86 servers, at least configured with 1 piece of local disk to store the temporary data. The size of memory is 64~256GB. The internet is Gigabit Ethernet or Infiniband. The hardware and software faults often occur;
➢ The GPU cluster system is mainly used for the migration. A computing node is configured with two CPU and two GPU.

Parallel Algorithm

The seismic data of each offset is split into the data block. Assume that the size of data block is *d*, the physical memory M of the computing nodes is enough, the size of data block meets: $m+d < M$. After the data has been split, the task can be indexed by the offset *O*, the bin block *C* and the data block *D*: $Task(O_i, C_j, D_k)$. Each task is independent and the task is sufficient to satisfy the requirements of the large scale parallelism.

```
Input:  Input traces, velocity, and aperture parameters;
Output:  Output traces.
Compute travel time table;
for all offsets par-do
    for all bin blocks par-do
        for all data blocks with this offset par-do
            for all output samples within the aperture do
                Lookup travel time table($t_s$ and $t_r$);
                Accumulate the input sample to an output sample;
            end for /*sample loop*/
        end for /*cell blocks loop*/
    end for /*data blocks loop*/
end for /*offset loop*/
Gather and write image.
```

Parallel Algorithm Implementation

➢ Inter-node parallelism

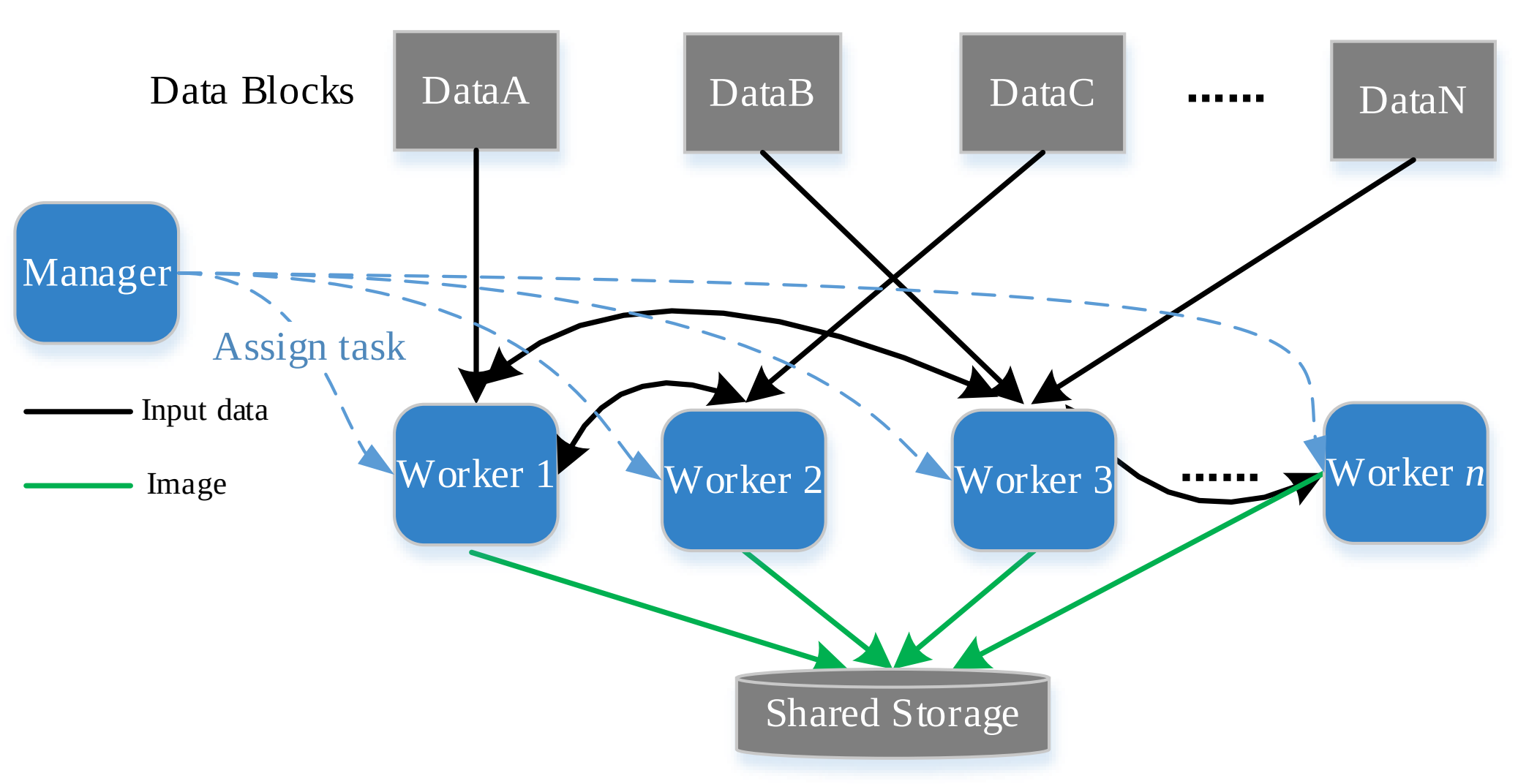

Fig 2. Tasks scheduling of inter-node

• **The first key technology of the algorithm implementation is the load balancing**. KPSDM adopts the "dynamic and asynchronous" task allocation policy to guarantee the load balancing. Each worker process maintains a task queue which contains only one task. Once the task is taken to be executed, the worker process requests a new task to the main process and placed it in the queue waiting to be executed. The stronger the computing ability of the node is, the higher the frequency of the task request is. Thus it is effective to assign the tasks according to the computing ability of the nodes. When the task is waiting in the queue, the seismic data and the travel time can be cached into the local storage in advance, achieving the asynchronous execution of the waiting task I/O and the computation of the current task.
• **The second key technology of the algorithm is to use local storage to extend the bandwidth of the consolidated stora**ge. The seismic data and the travel time corresponding to each task are stored in the local storage, and these data can serve other tasks which require the same data. The same data may exist on multiple computing nodes. The manager node records the load status of each node and chooses the idlest node to provide the data. Each computing node is not only the data providers, but also the data consumers. The two-way bandwidth of the network guarantees the scalability of KPSDM. The more the nodes more, the higher the I/O aggregate bandwidth is.
• **The third key technology of the algorithm is runtime fault tolerance**. For the large scale cluster system, the probability of the hardware or software fault of the computing node is very high. On the one hand the fault tolerance based on the checkpoint produces too much I/O consumption which is proportional to the scale of nodes. On the other hand restarting the entire application also brings the poor user experience. Therefore we developed a new parallel and distributed programming model (Geophysical Parallel Programming, GPP) instead of MPI. GPP is able to remove the application program when the program is running. The runtime system of GPP periodically checks the healthy status of the job process. Once some process fails, the ID of the failure process is reported to all the process by the event mechanism. The manager node of KPSDM is responsible for capturing fault event and reassigning all the tasks which has been assigned to the fault worker node and has not been executed.
• **The fourth key technology of the algorithm is how to process the slow nodes**. The slow nodes are also the common problems for the large scale cluster. Refer to the "task backup" policy of the MapReduce framework to solve the problem of the slow node which may make the running time of the job longer.

Parallel Algorithm Implementation(Continued)

➢ Intra-node parallelism

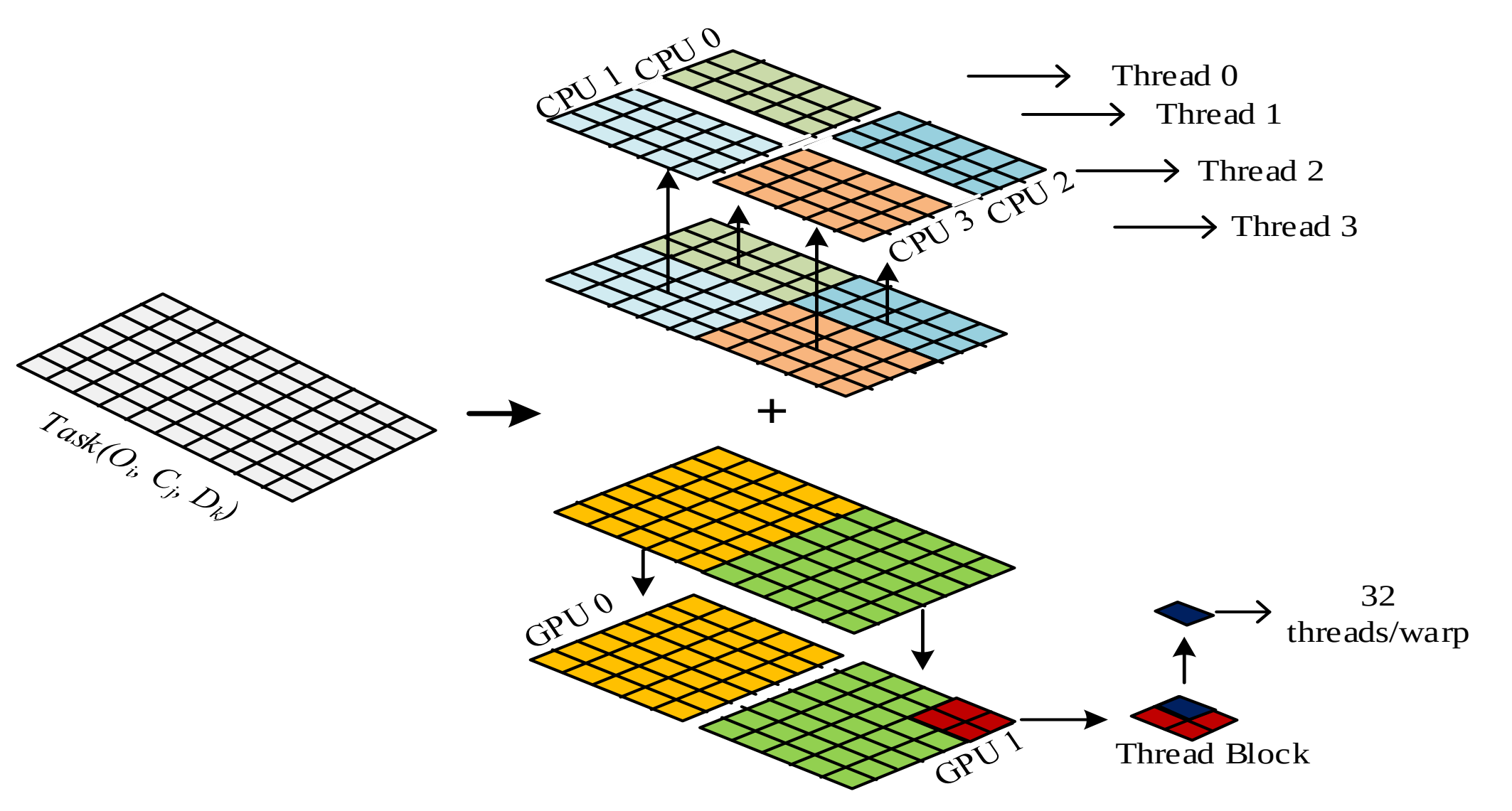

Fig 3. CPU-GPU cooperative computing

• **CPU and GPU do the same image task** and the seismic data is dynamically distributed between two types of processors. The stack of the intermediate result is output finally..
• **Each CPU core boots a thread**. The bin is split to the computing threads on average by using the round-robin way to ensure assigning the same amount of computing tasks to each thread. This method reduces the consumption for synchronization between threads. The input data is share by all threads which can enhance the hit rate of cache. Another important optimization is the vectorization in the aspect of the micro architecture. In the codes for production, there are much more conditional branches that are the biggest obstacle to optimize. A variety of code optimization method must be used for the code to remove the conditional branches. After the core code of KPSDM is fully vectorized, the whole performance increases more than one time.
• **The bins are shared by multiple GPU**, and the task allocation policy of GPU is the same with that of CPU. The task granularity needs to be further split for the GPU coprocessor. A part of the bin is assigned to each thread block. The output seismic trace of each bin is accomplished by 32 threads in one warp and every part of the output samples are computed by one thread. This task allocation policy in GPU not only can generate sufficient tasks but also can facilitate the aggregate access to the global memory. Loading the input seismic trace into the texture memory for faster access is one of the most critical optimization strategies. Other optimizations for KPSDM include: reducing the number of registers used for each thread to improve occupancy, hiding the delay of the data transfer from memory to video memory by CUDA streams.

Experimental

➢ The **cluster** for experiment is equipped with 256 computing nodes, each node is configured with the dual 8-core Intel E5-2670 CPU, 128GB memory and two 500GB HDD high speed local disk as RAID0. The network between the nodes is connected by the 50Gbps Infiniband. The shared storage uses the GPFS parallel file system. The test shows that the highest concurrent throughput is 4GB/s. When the throughput is more than about 2GB/s, the delay of reading increases significantly.
➢ Here we select a piece of **data** in the China northeast survey area. The total amount of data is 134GB, which contains 22,919,555 seismic traces. The number of each seismic trace is 79591.There is 73 offset totally. The amount of the travel time is 466GB. Each bin block contains 943 bins, and the amount of the corresponding travel time is 20GB. The size of the data block is 51GB.

Table 1. Data Sources at Run-Time

	Nodes	Read from Shared Storage/GB	Read from Local Storage/GB
Only seismic data is cached	32	4086.02	28988.25
	64	3190.51	28756.68
	96	3563.16	27499.04
	128	2941.49	28509.42
Both seismic data and travel time table are cached	**192**	**930.05**	**33051.98**
	237	**1055.27**	**32452.40**

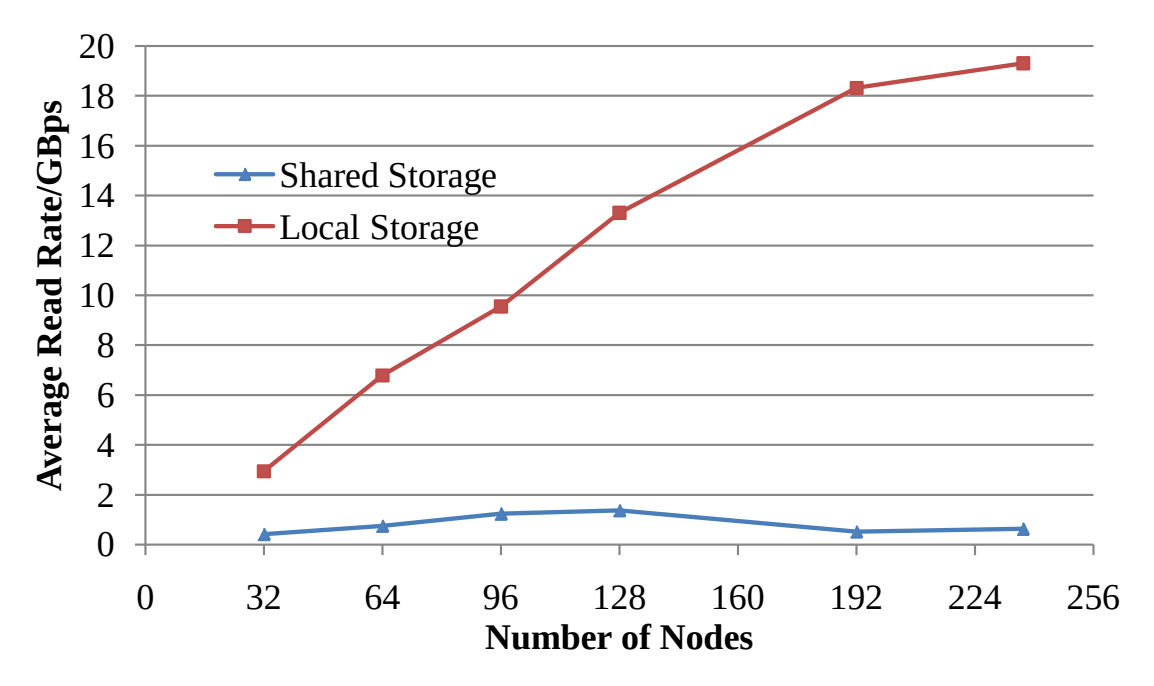

Fig 4. The average read rate of shared storage and local storage

➢ From Table 1 we can see that the job produced about 32TB input data during the run time. Compared to the original input data, it increased 53 times. When the cache mechanism of the travel time is not enabled, the local storage takes 90% of the **I/O** pressure. If the cache mechanism of the seismic data and the traveling table are both enabled, 97% of the I/O can be diverted.

➢ Because the distributed local storage provides a high aggregate bandwidth, the scalability of the KPSDM is very well, as shown in Figure 5, we even can get the super-linear **speedup**. The emergence of super-linear speedup is mainly because of the more load balance and the less reading latency with the increase of nodes. The reason for slowing down the accelerating is that the time of ending jobs accounted for the operation increases when the number of nodes is more than 192.

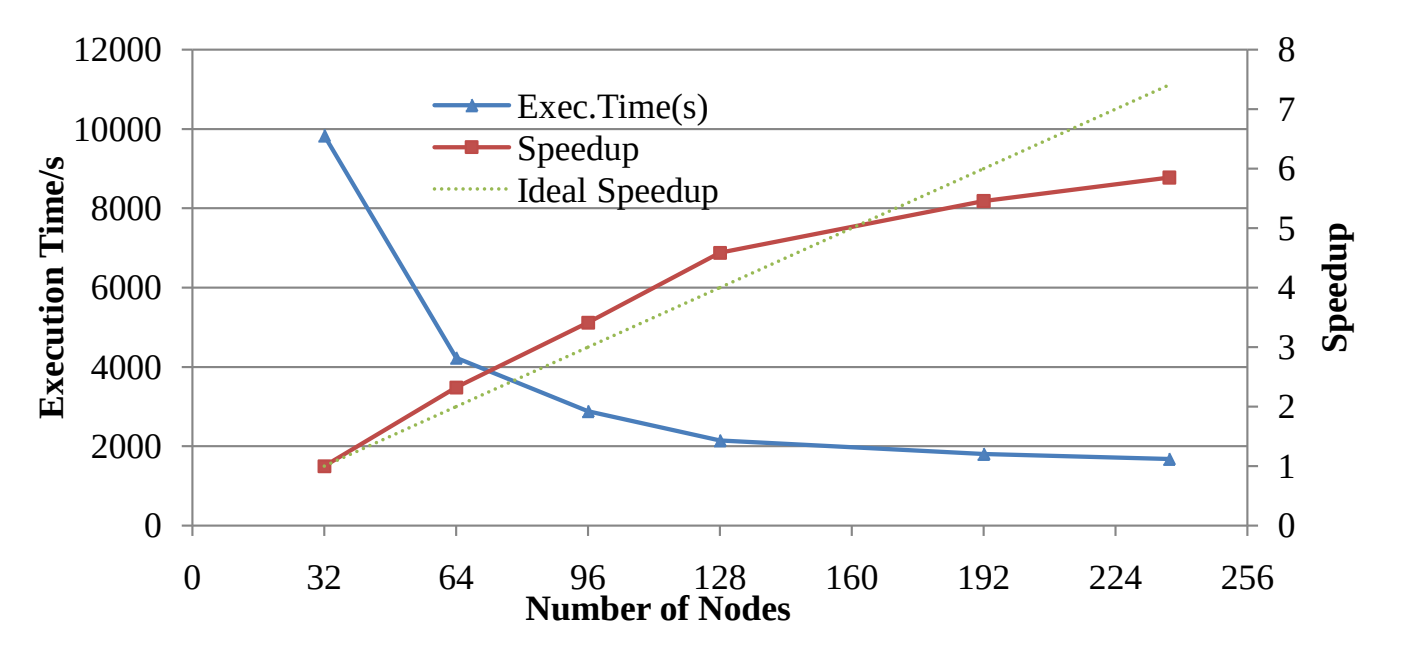

Fig 5. Execution time and speedup of inter-node parallelism

➢ The compute node contains 16 physical CPU cores, 32 hyper-threading. The test of thread-level **scalability** is shown in Figure 6. When the thread varies from 1 to 7, the speedup is approximate to linearity. When the thread changes from 8 to 16, there is a growing difference between the actual speedup and the theoretical speedup. The main reason for this situation is that each thread needs to read different travel time during the computing and the shared data between threads is very less. So the hit rate of L3 cache decreases with the increase of the threads. When the threads increase from 16 to 32, the program is still accelerated. The data in the figure shows that the performance of KPSDM obtains the gains of nearly 22% by the hyper-threading technology.

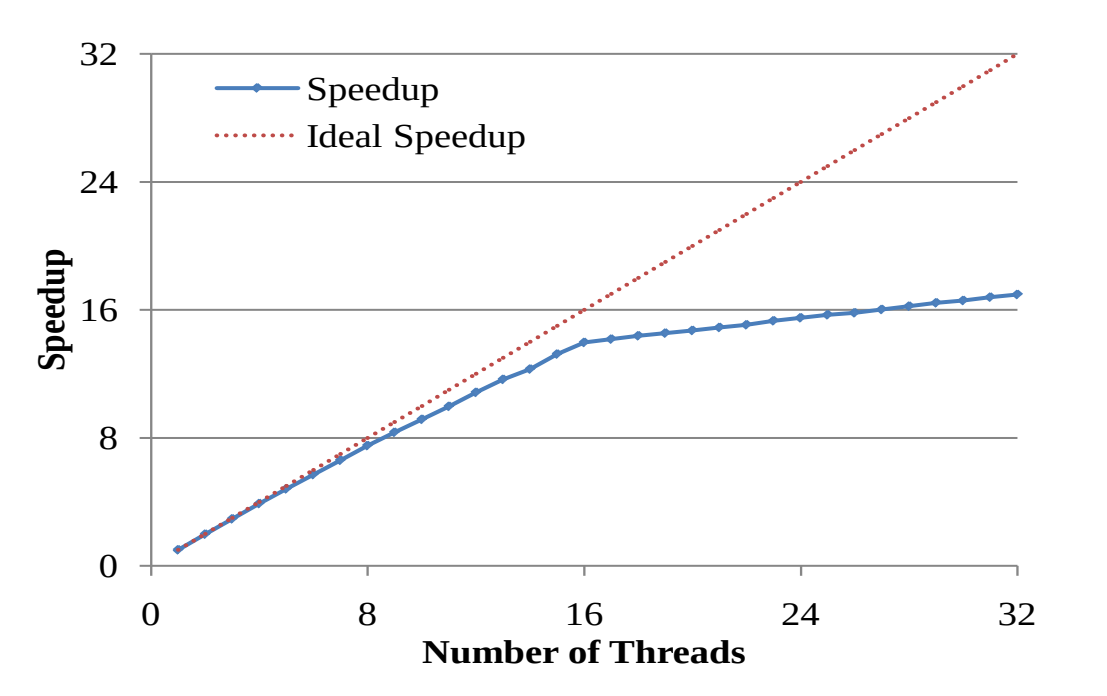

Fig 6. Speedup of intra-node parallelism

Conclusions

◆ The hybrid domain parallel KPSDM algorithm decomposed the task based on three domains including offset, imaging space, and seismic data. The algorithm eliminates the dependencies among tasks and facilitates the runtime fault tolerance and slow node processing. It is benefit to make full use of the computing resources of the large-scale cluster to make the load balance when the number of the tasks is sufficient. The task also can be spilt further based on the seismic trace to dynamically balance the load between CPU and heterogeneous processors.

◆ For a huge amount of seismic data and travel time, KPSDM hybrid domain parallel algorithm utilizes the local storage to cache data, which reduces the pressure of the shared storage. The aggregate bandwidth of the local storage provides strong guarantee for the scalability of the KPSDM algorithm.

◆ The architecture of computing system deeply affects the algorithm design of KPSDM, especially the storage architecture is essential to the scalability of KPSDM.