Supercomputing | DOI:10.1145/3481614

Processing Extreme-Scale Graphs on China's Supercomputers

BY YIMING ZHANG, KAI LU, AND WENGUANG CHEN

MANY APPLICATIONS, SUCH as Web searching, social network analysis, and power grid management, require extreme-scale graph processing. However, it is very challenging because graph processing exhibits unique characteristics such as more load imbalance, lack of locality, and access irregularity. The extreme graph scale makes the situation even worse.

Supercomputers have large compute, large memory, and fast interconnect. They seem ideal for extreme-scale graph processing. China has built several leading supercomputers in the world. For example, the Tianhe-2 and Sunway TaihuLight supercomputers have ranked No. 1 in Top500 list 10 times between June 2013 through May 2018. Like many other supercomputers, they use heterogeneous accelerators to achieve high performance for regular workloads such as stencil-based and structured grid-based computations. Are they also capable of processing extreme-scale graphs?

The TaihuLight supercomputer at the National Supercomputer Center in China's Jiangsu province.

In this article, we discuss our efforts to enable extreme-scale graph processing in two leading supercomputer architectures.

Tianhe hardware features. The Tianhe supercomputer has unique designs for its many-core CPU architecture and its high-performance interconnection network, providing both opportunities and challenges for graph processing.

One of the unique features of the TaihuLight machine is the heterogeneous on-chip SW26010 CPU.

Tianhe's computing nodes (CN) use the proprietary Matrix-2000+ CPUs (see Figure 1). A CN has three Matrix-2000+ CPUs, each of which has 128 2GHz cores. Each core has an in-order 8-to-12-stage pipeline extended with scalable vector extension (SVE). Matrix-2000+ CPUs adopt a regional autonomous parallel architecture where one CPU is composed of four regions connected through a scalable on-chip communication network. Each region is a functionally independent supernode (SN) with four panels communicating through an intra-region interface. Each panel has eight cache-coherent compute cores. SVE enables Matrix-2000+ to choose the most appropriate vector length via two usage modes: auto vector-length agnostic (AVLA) mode and assembly vector-length specified (AVLS) mode. AVLA mode can automatically pack sub-vectors into vectors but requires synchronization between processing of two vectors, while AVLS mode allows programmers to specify user-defined sub-vector lengths.

Tianhe's network subsystem adopts a multi-dimensional tree topology with optoelectronic-hybrid interconnection, which combines the benefits of both tree and n-D-Torus topologies. The networking logic is integrated into

PHOTO BY FEATURE CHINA/BARCROFT MEDIA VIA GETTY IMAGES the network interface chip (HFI-E) and the network router chip (HFR-E). HFI-E implements the proprietary MP/RDMA (mini packet/remote direct memory access) communication and collective offloading mechanism. CNs connected to an HFR-E are in the same communication domain. Tianhe has highly optimized its intra-domain communication, which is an order of magnitude faster than its inter-domain communication crossing multiple HFR-Es.

Leveraging hardware features for graph processing. Different from the traditional SIMD (single-instruction-multiple-data) technique, the Matrix2000+ CPUs support vectorization to accelerate graph computation. We leverage SVE to realize efficient graph traversal.

Traditional vectorization induces synchronization between processing consecutive vectors (by inserting stalls) and thus lowers the overall performance. Fortunately, we find that graph traversal (such as BFS) simply scans a vertex range to determine the vertices to-be-traversed at the next level, allowing avoiding synchronization if none of the vertices belongs to more than one level. This situation might exist only because of the existence of loops in the graph. To address the loop problem, if a vertex exists in two successive levels and causes a loop, we split it into two virtual ones. Moreover, if we find a vertex that belongs to multiple levels during pre-processing, we use a virtual vertex at each level to participate in that level's vectorized processing.

We adopt AVLA and AVLS to realize graph traversal efficiently. If the traversal is likely to encounter loops (for example, in top-down BFS), then we adopt AVLA to automatically pack unvisited neighbor vertices (sub-vectors) into vectors while avoiding vertex splitting (at the cost explicit synchronization). Otherwise, it is unlikely to encounter loops (for example, in bottom-up BFS), and thus we pack the neighbors into the vectors through AVLS and split vertices once loops occur. AVLA and AVLS accelerate the procedure that every thread (core) handles a different vertex range and examines the edges connected to unvisited vertices, so as to determine whether the neighbor vertices should be visited on the next level.

To adapt graph processing to the topology of Tianhe's multi-dimensional tree network, we refactorize the graph with *fusion* and *fission*[3] when storing graph vertices and edges. Specifically, fusion organizes a set of neighboring low-degree vertices into a super-vertex (for processing locality), and fission splits a high-degree vertex into a set of sibling sub-vertices (for load balancing). Refactorization is performed in parallel by all workers on CNs, and we resolve the potential conflict by double-checking on vertex states. A worker checks whether the vertex has been processed both before moving it to the processing queue and before performing fusion/fission. If a conflict occurs, further processing will be skipped.

The vertices and edges of the refactorized graphs are assigned to the CNs in the network according to the proximity of the multidimensional tree topology. Neighboring vertices are assigned to the same communication domain. Graph refactorization could be pipelined with assignment and thus only induces a small extra overhead.[3] Note the partitioning results are reusable. Thus, it is worth paying for the extra refactorization overhead in most cases.

We further leverage the topology information to perform aggressive message aggregation. Messages are gathered to the responsible nodes (referred to as monitors) in the source domains, transferred between monitors, and scattered to the target nodes in the target domains. We adopt adaptive buffer switching and dynamic buffer expansion to reduce communication cost effectively.

Applications of graph processing on Tianhe. The Tianhe graph processing system has been widely used in industry throughout China in areas such as computational biology, industrial simulation, and visualization.

Beijing Genomics Institute (BGI) and Shanghai Institute of Materia Medica (SIMM) jointly constructed a high-throughput drug virtual screening platform based on Tianhe graph processing system. The platform screened over 40 million molecular compounds for anti-Ebola virus drugs per day, achieves the fastest high-throughput virtual drug screening in history, and plays an im-

The Tianhe supercomputer has unique designs for its many-core CPU architecture and its high-performance interconnection network, providing both opportunities and challenges for graph processing.

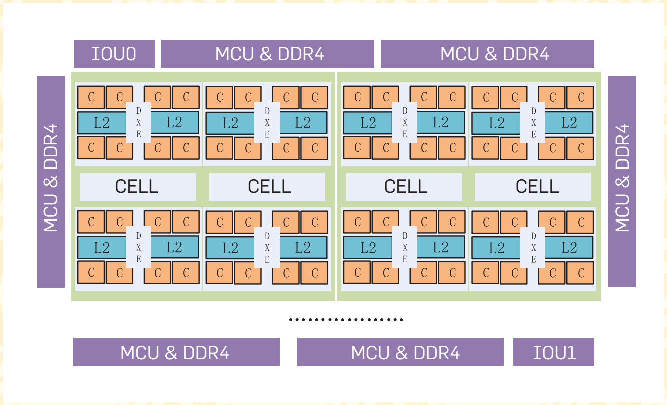

Figure 1. Matrix-2000+ CPU architecture.

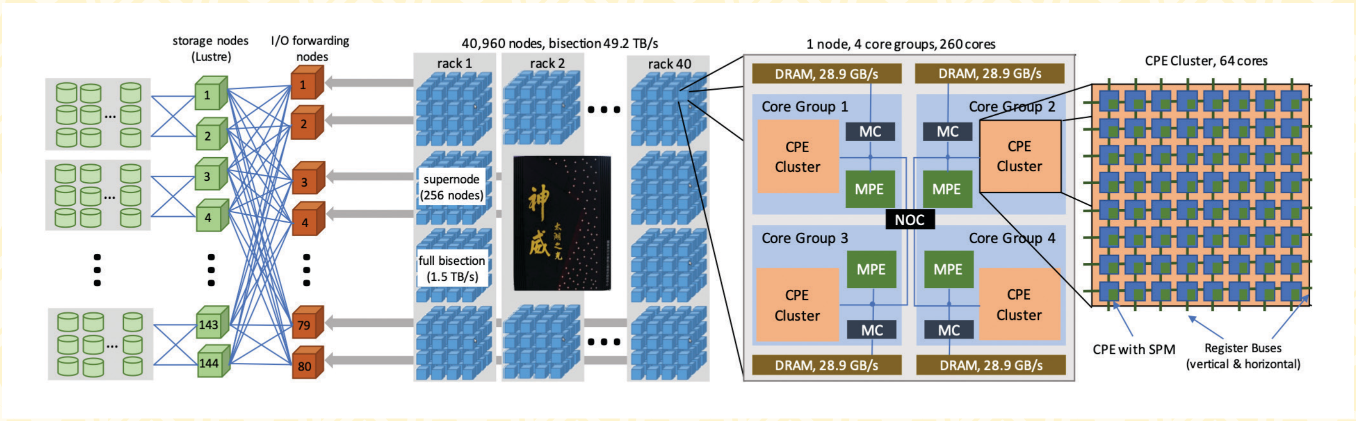

Figure 2. The architecture of TaihuLight.

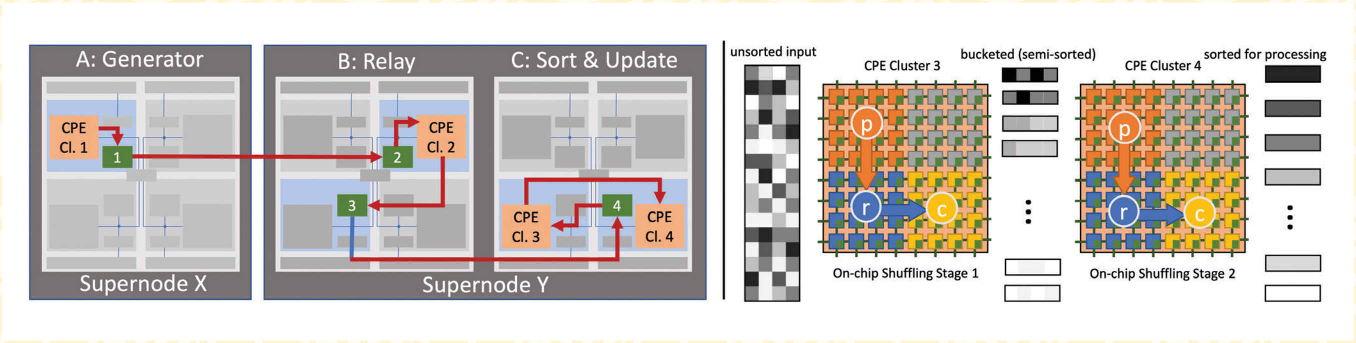

Figure 3. Supernode routing and module mapping in CPEs.

portant role in anti-Ebola drug development. The core algorithm, Lamarckian Genetic Algorithm (LGA),[2] is formed as a global-local-hybrid search problem and deeply accelerated on Tianhe graph processing system using 512 ~8192 CNs (up to 19.7K CPU cores). The efficiency of our graph processing system reaches as high as 60%.

Shaanxi Key Laboratory of Large-scale Electromagnetic Computing (LEC) has developed the complex matrix local block pivoting LU (LBPLU) decomposition software for applications of massive parallel Method of Moments (MoM). The LEC laboratory ran LBPLU with local pivoting on Tianhe to solve the matrix equation generated by MoM. Specifically, for simulating the electromagnetic scattering of an aircraft, LBPLU divides the aircraft surface into a grid structure, where each grid node is a vertex of a graph, and the influence between grid nodes is modeled as edges. LBPLU performs triangulation on the grid to realize flow visualization. When running LBPLU, our system achieves as high as 50.16% parallel efficiency when the parallel scale is 8192CNs (19.7K cores).

To maximize utilization of the full-bisection intra-supernode bandwidth, we form target groups using supernode boundaries.

We have also deployed the graph processing system on a subset of the next-generation exascale Tianhe supercomputer, which consists of 512CNs with 96608 cores. Performance evaluation shows that the 512-node sub-system achieves 2131.98 Giga TEPS (traversed edges per second) for the BFS test of Graph500, which outperforms Tianhe-2 Supercomputer with 16x more nodes.

Processing Graphs on Sunway TaihuLight

The architecture of TaihuLight is illustrated in Figure 2. One of the unique features of TaihuLight machine is the heterogeneous on-chip SW26010 CPU (see right part of Figure 2). Each SW26010 CPU comprises four core groups (CGs) connected via a low-latency on-chip network (NoC). Each CG consists of a management processing element (MPE), a 64-core computing processing element (CPE) cluster, and a memory controller (MC), and thus a total of 260 cores per CPU (node). Each CPE comes with a 64KB scratch pad memory (SPM) without cache, which requires explicit programmer control. The architecture demands manual coordination of all data movement, which is a particularly challenging task for irregular random accesses.

The TaihuLight CNs are connected via a 2-level InfiniBand network. A single-switch with full bisection bandwidth connects all 256 nodes within a supernode, while a fat-tree with 1/4 of the full bisection bandwidth connects all supernodes (see left part of Figure 2).

Mapping the graph processing modules to heterogeneous processors. Within each SW26010 CPU, the four CGs are assigned with distinct functions as shown in Figure 3: (A) Generation, (B) Relay, (C1) Coarse sort, and (C2) Update. This function mapping is static, and each function is performed by one CG only. The goal of this mapping is to achieve balanced CG utilization. This pipelined architecture allows us to process batched data in a streaming way, gaining lower I/O complexity to main memory and higher utilization of the on-chip bandwidth.

At the second level of specialization, we leverage the specific hardware features within each CG. The MPE is well suited for task management, plus network and disk I/O, while the CPEs are tightly connected through the 2D fast communication feature, naturally leading us to assign communication tasks on the MPE and data sorting tasks on the CPEs.

Supernode routing. This technique targets efficient inter-node communication to enable our heterogeneous processing pipeline on the full system.

The performance of distributed graph applications are usually damaged by large numbers of small messages sent following the graph topology. The all-to-all style small messages among 10,000s of nodes is inefficient due to per-message overheads (routing information, connection state, and so on.) We propose a supernode routing technique to mitigate this by factoring all compute nodes into groups according to their supernode affiliation. Each node combines all messages to nodes within the same target group into a single message sent to a designated node within that group. This so-called relay node unpacks the received messages, combing messages from different source groups, repack the messages to each in-group target node into one message, and distributes them to appropriate peers.

To maximize utilization of the full-bisection intra-supernode bandwidth, we form target groups using supernode boundaries. Each source node minimizes the number of relay nodes it sends to within a target group (usually one relay node per target group) to perform message aggregation effectively. To achieve load balance, each node in a target group acts as a relay node. The situation is more complicated if there are failure nodes in a supernode, and we use a stochastic replay assignment to maintain load balance.

The Shentu graph processing framework. In addition to the hardware specialization and supernode routing, we also have other innovations such as on-chip CPU sorting and degree-aware messaging. We omit them here due to limited space. Readers interested in more details should refer to Lin et al.[1]

Finally, we designed and implemented a vertex-centric graph processing framework, Shentu, in Sunway TaighuLight. It could support graph algorithms such as PageRank, WCC, and BFS with around 30 lines of code to run on the full system of TaighuLight.

It should be noted that the Sogou graph is the largest real graph processed in literature, which has 12 trillion edges and is prohibitive for small scale systems. Shentu could process it with 8.5s for each iteration of PageRank in full scale Sunway TaighuLight (see the accompanying table). The 42.kron (70 trillion edges) is also the largest synthetic graph processed in literature to date.

Conclusion

In this article, we showed how graph processing is efficiently supported by supercomputers with different heterogeneous architecture characteristics: on-chip processing element array with SPMs and wide vector units. We also showed how techniques such as vectorization and supernode routing are used to optimize the all-to-all messages of graph computing. We expect the result not only enables extreme-scale graph processing, but also hints at the possible fusion of supercomputing and big data architectures. C

						Aggregated GPEPS					
	#Vtx	#Edge	Size	#Node	IO cost(s)	PR	BFS	WCC	BC	KCore	Net. Usage
twitter	41.6m	1.4b	16GB	16	37.2	0.75	0.36	0.54	0.30	0.01	28.1%
uk-2007	105.8m	3.7b	41GB	64	28.9	3.99	3.36	3.97	0.87	0.13	28.1%
clueweb	978.4m	42.5b	476GB	100	119.4	5.04	2.47	4.79	1.23	0.41	47.4%
uk-2014	747.8m	47.6b	532GB	100	132.7	4.67	2.06	3.81	0.35	0.32	47.1%
weibo	349.9m	42.4b	474GB	100	121.3	8.22	2.56	7.15	4.77	0.72	40.8%
hyberlink	3.5b	128.7b	1.5TB	256	243.0	17.20	3.68	14.49	0.32	0.26	60.1%
sogou	271.8b	12253.9b	136.9TB	38656	2130.6	1443.4	30.8	224.4			18.8%
34.rmat	17.1b	274.8b	3.3TB	1024	-	83.0	34.5	77.7	44.8	0.25	29.2%
34.erdos	17.1b	274.8b	3.3TB	1024	-	52.5	48.8	51.3	40.9	18.60	63.7%
34.kron	17.1b	274.8b	3.3TB	1024	-	72.8	18.2	62.3	43.1	0.82	40.2%
42.kron	4398.0b	70368.7b	844.4TB	38656	-	1984.8	773.9	1956.0			40.0%

Dataset and performance of Shentu.

References

1. Lin, H., et al. ShenTu: Processing multi-trillion edge graphs on millions of cores in seconds. In *Proceedings of 2018 ACM Intern. Conf. on Supercomputing.*
2. Morris, G., Goodsell, D., Halliday, R.S., Huey, R., Hart, W., Belew, R., and Olson, A. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. *J. Comput. Chem. 19* (1998), 1639–1662.
3. Zhang, Y., Wang, H., Jia, M., Wang, J., Li, D-S., Xue, G. and Tan, K. TopoX: Topology refactorization for minimizing network communication in graph computations. *IEEE/ACM Trans. Networking 28* (2020), 2768–2782.

Yiming Zhang is a professor at National University of Defense Technology in Changsha, China.

Kai Lu is a professor at National University of Defense Technology in Changsha, China.

Wenguang Chen is a professor at Tsinghua University in Beijing, China.