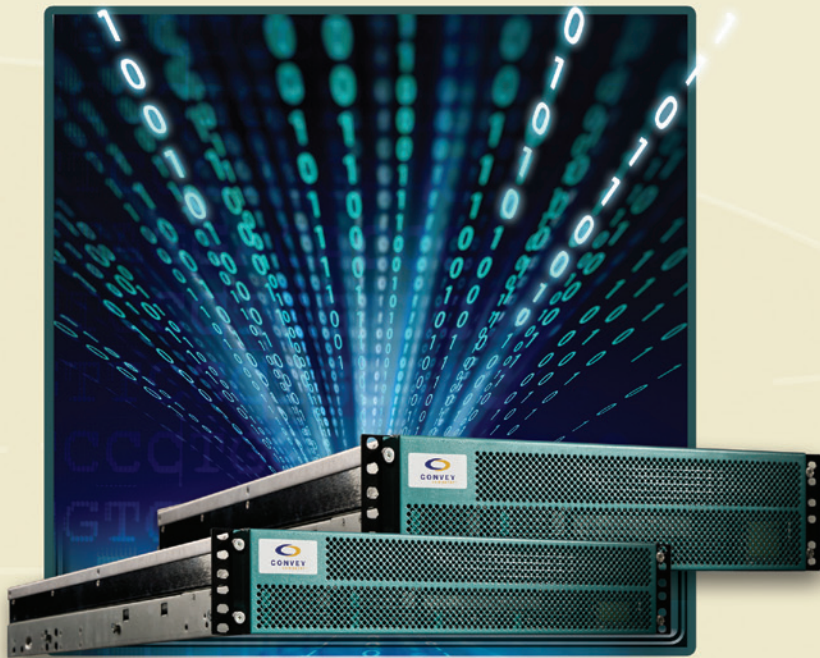
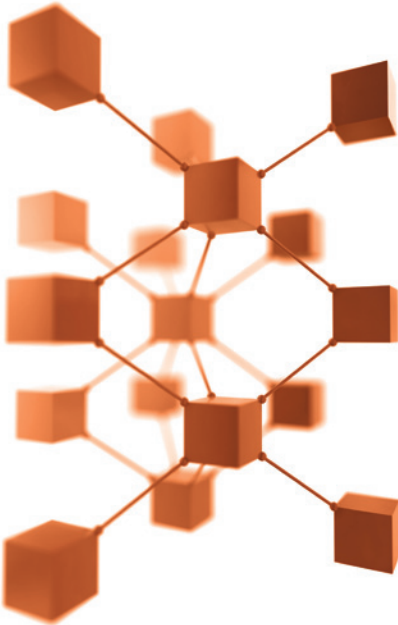


# Hybrid-core: The “Big Data” Computing Architecture





#### Contents

- 1 Introduction
- 1 Data-intensive problems
- 2 Graph algorithms
- 2 Architectural requirements
- 3 The Graph 500 benchmark
- 4 Hybrid-core computing and the Graph 500 benchmark
- 5 Performance results
- 5 Conclusions
- 6 Appendix A. November, 2011 Graph500 benchmark results

# Hybrid-core: The “Big Data” Computing Architecture

**High-performance computing (HPC) is no longer just numerically intensive, it’s now data-intensive—with more and different demands on HPC system architectures.**

## Introduction

Over the past few decades, the explosion of worldwide technology has led to a corresponding explosion in the amount of data that is “in your face.” Cell phones, personal computers, tablets, audio and video recording devices, social media interaction—all of these add to the storm of bits swarming around us every day.

An obvious question is “is there opportunity here?” It certainly seems like there should be. For example, if I’m an advertiser on Facebook, and someone has requested additional information from a placed advertisement, it makes sense to advertise similar products to all the friends, and friends of friends, of the information requester. Or, as a bioinformatician, you have 20 billion short-read sequences obtained from a DNA sequencing instrument, and need to assemble them into a single genome. If you’re successful, you can create the next drought-tolerant corn plant.

The point is, if we can capture, manage, analyze, and understand all of this data, there is an opportunity to increase research quality, design better products, and streamline otherwise inefficient business processes. And it’s not just the volume of data that’s the challenge—it’s analyzing, visualizing and understanding the *relationships* between elements of data.

## Data-intensive problems

### *A new class of problems*

While there is no precise definition of “data intensive,” these kinds of problems can certainly be identified when data is the primary challenge, whether it is the complexity, size, or rate of the data acquisition.<sup>1</sup> Attributes generally include:

- Large data sets (multi-terabyte, petabyte, exabyte, zettabyte, etc.)
- High ratio of memory accesses to computing
- Extremely parallelizable read accesses/computing
- Highly dynamic data—may often be processed in realtime

Obviously, this covers a lot of ground. For the most part, a specific algorithm type—*graph algorithms*—represent many of the general characteristics of data-intensive computing applications.

<sup>1</sup> <http://dicomputing.pnnl.gov>

## Graph algorithms

To narrow down the scope of this paper (and provide some supporting quantitative analysis), we will examine one subset of data-intensive problems: Those that use some kind of graph data structures. Graphs provide a flexible mechanism for describing relationships between data elements. They are among the most ubiquitous models of both natural and human-made structures, and can be used to model many types of relations and process dynamics in physical, biological, and social systems.<sup>2</sup>

Certainly graph problems aren't new. Modeling data relationships using graphs in software and hardware is "reasonably" easy. Data structures can be constructed that contain information about specific nodes in the graph, connections and relationships to other nodes, and so on.

However, using graph algorithms to model real-world relationships between data, and managing graphs that have billions of nodes and edges requires multiple terabytes of storage, *is* new (and "different"). Computer architectures that effectively execute these algorithms are also new; as the National Science Foundation states: "Data intensive computing demands a fundamentally different set of principles than mainstream computing."<sup>3</sup>

Let's look at some of the architectural requirements for computing systems that can efficiently tackle problems with these attributes.

## Architectural requirements

As preparation to developing an inventory of architectural "wants" to solve these kinds of problems, let's start with examining how today's conventional commodity computer systems<sup>4</sup> handle (or don't handle) graph algorithms.

*Existing HPC architectures are designed for simulation and modeling—they're great for floating-point operations, but not for graph traversal. The new HPC demands different architectures.*

### *Deficiencies of commodity systems*

In general, commodity systems have multiple difficulties in managing data processing on such a scale:

*Very little parallelism.* The number of concurrent operations in a typical commodity server is limited to the number of CPU cores—on the order of eight to thirty-two. In order to get thousands or tens of thousands of concurrent operations, huge clusters of systems are needed. Limited parallelism means low CPU utilization and execution time becomes dominated by memory latency.

*Memory architecture poorly mapped to the type of memory accesses.* The memory architecture built-into today's commodity processors is cache-line oriented (64 bytes). Whenever a data structure is requested from memory (let's say an 8-byte doubleword), all 64 bytes are transferred from memory and brought into cache. Essentially, only eight out of sixty-four bytes (12.5%) of the system's memory bandwidth is usable. This hurts performance when data accesses exhibit poor locality (as is typical in graph problems).

*Lack of synchronization primitives.* In a parallel programming environment, modification of data structures must be coordinated to eliminate contention. That coordination needs to have minimal impact on execution time—especially with high thread counts. Typically, software (e.g. the operating system) manages accesses to memory, and introduces significant execution time overhead.

### *Desirable architectural features*

What type of architectural features are desirable in a computer system that executes graph algorithms? Following are the features that give the most performance for the least cost/space/power:

<sup>2</sup> [http://en.wikipedia.org/wiki/Graph\\_theory#Problems\\_in\\_graph\\_theory](http://en.wikipedia.org/wiki/Graph_theory#Problems_in_graph_theory)

<sup>3</sup> [http://www.nsf.gov/funding/pgm\\_summ.jsp?pims\\_id=503324&org=IIS](http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503324&org=IIS)

<sup>4</sup> The term "commodity computer system" refers to "pizza-box" or blade systems based on standard x86 processors and chipsets.

*Balance between compute elements and memory subsystem performance.* Most data-intensive problems require minimal compute resources (especially in terms of floating operations), and require more memory subsystem performance. Ideally, as in a reconfigurable or hybrid-core computing system, the compute elements can be changed on-the-fly to adapt to the application's compute needs.

*Memory subsystem designed for random access.* A memory system that returns exactly what was requested will maximize the available memory bandwidth.

*High bandwidth memory subsystem.* In addition to optimizing bandwidth, the aggregate bandwidth should be as high as possible. In addition, many thousands of simultaneous outstanding requests should be supported.

*Massive multi-threaded capability.* A combination of compute and memory requirements, the ability to support tens or hundreds of thousands of concurrent execution threads is a requirement. More parallelism reduces time-to-answer, improves hardware utilization, and increases efficiency.

*Hardware-based synchronization primitives.* With high degrees of parallelism comes the challenge of synchronizing read/write access to memory locations. Data integrity demands that a read-modify-write operation to a memory location is an indivisible operation. When the synchronization mechanism is "further away" from the operation, more time is spent waiting for the synchronization, with a corresponding reduction in efficiency of parallelization. Ideally, synchronization is implemented in hardware in the memory subsystem.

## The Graph 500 benchmark

As described previously, today's commodity servers, as well as systems designed specifically for numerically intensive algorithms ("supercomputers"), are ill suited for most applications in the data-intensive sciences. Similarly, attempts to characterize system performance using popular benchmarks for these kinds of systems (e.g. the Top500) will not yield representative results.

*"Current benchmarks and performance metrics do not provide useful information on the suitability of supercomputing systems for data intensive applications. A new set of benchmarks is needed in order to guide the design of hardware architectures and software systems intended to support such applications and to help procurements."*  
—www.graph500.org

Recognizing the need for a benchmark suite that will more accurately measure performance on graph-type problems, a steering committee of HPC experts from academia, industry, and national laboratories created the Graph500 benchmark. Currently the benchmark constructs an undirected graph and measures the performance of a kernel that performs a breadth-first search of graph.<sup>5,6</sup>

As shown in Figure 1, the benchmark has multiple data set sizes ("scale").<sup>7</sup>

---

Problem class	Scale	Edge factor	Approx. storage size in TB
Toy (level 10)	26	16	0.0172
Mini (level 11)	29	16	0.1374
Small (level 12)	32	16	1.0995
Medium (level 13)	36	16	17.5922
Large (level 14)	39	16	140.7375
Huge (level 15)	42	16	1125.8999

---

Figure 1. Data set sizes ("Scale") of the Graph500 benchmark.

<sup>5</sup> <http://www.graph500.org/index.html>

<sup>6</sup> [http://en.wikipedia.org/wiki/Breadth-first\\_search](http://en.wikipedia.org/wiki/Breadth-first_search)

<sup>7</sup> <http://www.graph500.org/specifications.html>

In addition, incremental scale sizes are allowed by appending a “+” or “++,” e.g. “Mini++” is scale 31. The performance metric of the benchmark is TEPS (Traversed Edges Per Second). The results are then sorted *first by problem size, then by performance* (see Appendix A).

The kernel of the breadth-first search portion of the benchmark (Figure 2) contains multiple constructs that are common to many graph-type algorithms—specifically a high degree of parallelism, and indirect (or “vector of indices”) memory references.

```

for (k = k1; k < oldk2; ++k) {                                ← Parallelize multiple re
const int64_t v = vlist[k];
int64_t vo;
for (vo = xoff[2*v]; vo < xoff[2*v+1]; ++vo) {
const int64_t j = xadj[vo];
if (bfs_tree[j] == -1) {                                     ← bfs_tree[xadj[vo]] is v
int64_t t = readfe (&bfs_tree[j]);
if (t == -1) {
t = v;
vlist[int_fetch_add (&k2, 1)] = j; ← Single Writer, Critical
}
writeef (&bfs_tree[j], t);
}
}
}

```

Figure 2. The kernel of the breadth-first search algorithm extracted from the Graph500 release code.

### Hybrid-core computing and the Graph 500 benchmark

As we have reviewed, solving graph problems takes a different approach to computing. One such approach is the Convey HC (hybrid-core) family of computer systems.<sup>8</sup> The Convey systems offer a balanced architecture: reconfigurable (via Field Programmable Gate Arrays—FPGAs) compute elements, and a supercomputing-inspired memory subsystem (Figure 3).

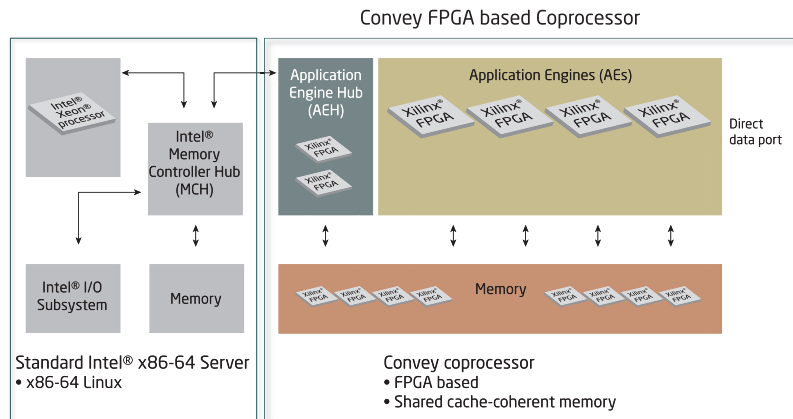


Figure 3. Overview of the Convey hybrid-core computing architecture.

The benefit of hybrid-core computing is that the compute-intensive kernel of the Graph500 breadth-first search is implemented in hardware on the FPGAs in the coprocessor. The FPGA implementation allows much more parallelism than a commodity system (the Convey memory subsystem allows up to 8,192 outstanding concurrent memory references). The increase in parallelism combined with the hardware implementation of the logic portions of the algorithm allow for increased overall performance with much less hardware.

<sup>8</sup> <http://www.conveycomputer.com/technology.html>

In addition to increased parallelism, the memory subsystem of the Convey systems is specifically designed to provide high bandwidth for parallel references that exhibit poor locality (e.g. offers high performance for random accesses). Thus, the vector of indices portion of the code is highly accelerated over architectures that are not well-suited for random accesses.

## Performance results

The hybrid-core architecture of the Convey hybrid-core family lends itself exceedingly well to the Graph500 benchmark (Figure 4). While the problem size is considered “small” (which is understandable, given that the benchmark is run on a single node system), the performance-per-watt and performance-per-dollar are well beyond any other system on the list.

Figure 4 is a partial list (problem scale 28-31) of the performance results for the November 2011 release of the Graph500 benchmark. Approximate power requirements allow for an arbitrary metric illustrating power efficiency (MTEPS/kw).

*“Pound-for-pound, watt-for-watt, the Convey hybrid-core systems provide more graph processing power than any other system on the Graph500 list—hands-down.”*

Rank	System	Site	Scale	MTEPS	MTEPS/ kW
20	Dingus (Convey HC-1ex - 1 node / 4 cores, 4 FPGAs)	SNL	28	1,759	2,345
22	Vortex (Convey HC-1ex - 1 node / 4 cores, 4 FPGAs)	Convey Computer	28	1,675	2,233
23	Convey01 (Convey HC-1ex - 1 node / 4 cores, 4 FPGAs)	Bielefeld University, CeBiTec	28	1,615	2,153
24	HC1-d (Convey HC-1 - 1 node / 4 cores, 4 FPGAs)	Convey Computer	28	1,601	2,135
25	Convey2 (Convey HC-1 - 1 node / 4 cores, 4 FPGAs)	LBL/NERSC	28	1,598	2,130
7	IBM BlueGene/Q (512 nodes)	IBM Research, T.J. Watson	30	56,523	1,809
40	ultraviolet (4 processors / 32 cores)	SNL	29	420	420
9	SGI Altix ICE 8400EX (256 nodes / 1024 cores)	SGI	31	14,085	363
26	IBM and ScaleMP/vSMP Foundation (128 cores)	LANL	29	1,551	242
32	IBM and ScaleMP/vSMP Foundation (128 cores)	LANL	30	1,036	162
14	DAS-4/VU (128 processors)	VU University	30	7,135	139
34	Hyperion (64 nodes / 512 cores)	LLNL	31	1,004	39
43	Kraken (6128 cores)	LLNL	31	105	21
35	Matterhorn (64 nodes)	CSCS	31	885	18
19	Todi (176 AMD Interlagos, 176 NVIDIA Tesla X2090)	CSCS	28	3,060	17
42	Knot (64 cores / 8 processors)	UCSB	30	177	9
49	Gordon (7 nodes / 84 cores)	SDSC	29	30	3
33	Jaguar (224,256 processors)	ORNL	30	1,011	0

Figure 4. Performance and power on the Graph500 benchmark (for problem size 28-31).<sup>9</sup>

## Conclusions

The massive explosion of data available for analysis and understanding is creating a “whole new HPC,” with demands on existing HPC architectures that cannot be fulfilled by current commodity systems. Future generations of HPC systems will be required to acknowledge some of the architectural requirements of data-intensive algorithms. For example, memory subsystems will need to increase effective bandwidth, more parallelism will be needed, and synchronization primitives will need to be “closer” to the memory subsystem.

By implementing a balanced, hybrid approach, Convey’s hybrid-core family of systems are able to execute problems in the data-intensive sciences much more effectively. The hybrid-core architecture is poised for exascale levels of computing in the data-intensive sciences because it offers reconfigurable compute elements balanced with a supercomputer-inspired memory subsystem.

<sup>9</sup>One entry was removed (#17) because it employed a different BFS algorithm.

## Appendix A. November, 2011 Graph500 benchmark results<sup>10</sup>

### Complete Results - November 2011

Rank	Machine	Owner	Problem Size	TEPS	Implementation
1	NNSA/SC Blue Gene/Q Prototype II (4096 nodes / 65,536 cores)	NNSA and IBM Research, T.J. Watson	32	254,349,000,000	Custom
2	Hopper (1800 nodes / 43,200 cores)	LBL	37	113,368,000,000	Custom
2	Lomonosov (4096 nodes / 32,768 cores)	Moscow State University	37	103,251,000,000	Custom
3	TSUBAME (2732 processors / 1366 nodes / 16,392 CPU cores)	GSIC Center, Tokyo Institute of Technology	36	100,366,000,000	Custom
4	Jugene (65,536 nodes)	Forschungszentrum Jülich	37	92,876,900,000	Custom
5	Intrepid (32,768 nodes / 131,072 cores)	ANL	35	78,869,900,000	Custom
6	Endeavor (Westmere) (320 nodes / 640 sockets / 3840 cores)	Parallel Computing Lab / Intel Labs	32	57,567,900,000	Custom/Intel
7	IBM BlueGene/Q (512 nodes)	IBM Research, T.J. Watson	30	56,523,400,000	Custom
8	Franklin (4000 nodes / 16,000 cores)	LBL	36	19,955,100,000	Custom
9	SGI Altix ICE 8400EX (256 nodes / 1024 cores)	SGI	31	14,085,400,000	Reference
10	SGI Altix UV 1000 (2048 cores)	SGI	32	10,161,300,000	Reference
11	Red Sky (512 nodes / 4096 cores)	SNL	33	9,470,000,000	Custom/ETI SWARM
12	Endeavor (Sandy Bridge) (256 nodes / 4096 cores)	Intel	34	9,250,000,000	Custom/ETI SWARM
13	Lonestar (1024 nodes / 12 cores)	TACC	35	9,240,000,000	Custom/ETI SWARM
14	DAS-4/VU (128 processors)	VU University	30	7,135,130,000	Reference MPI/OpenMP
15	BlueGene/P (2048 nodes / 8192 cores)	Moscow State University	32	6,930,560,000	Custom
16	Jaguar PF (512 nodes / 1024 cores)	ORNL	33	6,260,000,000	Custom/ETI SWARM
17	mirasol (Quad-Socket Intel E7-8870 - 2.4 GHz - 10 cores / 20 threads per socket)	Georgia Tech	28	5,125,652,789	Custom
18	Blacklight (512 processors)	PSC	32 (Small)	4,452,270,000	Custom
19	Todi (176 AMD Interlagos, 176 NVIDIA Tesla X2090)	CSCS	28	3,059,970,000	Custom GPU Result
20	Dingus (Convey HC-1 <sup>™</sup> - 1 node / 4 cores, 4 FPGAs)	SNL	28	1,758,682,718	Convey Custom
21	Wingus (Convey HC-1 <sup>™</sup> - 1 node / 4 cores, 4 FPGAs)	SNL	27	1,754,693,945	Convey Custom
22	Vortex (Convey HC-1 <sup>™</sup> - 1 node / 4 cores, 4 FPGAs)	Convey Computer Corporation	28	1,675,401,731	Convey Custom
23	Convey01 (Convey HC-1 <sup>™</sup> - 1 node / 4 cores, 4 FPGAs)	Bielefeld University, CeBITec	28	1,614,891,176	Convey Custom
24	Hc1-d (Convey HC-1 - 1 node / 4 cores, 4 FPGAs)	Convey Computer Corporation	28	1,601,346,927	Convey Custom
25	Convey2 (Convey HC-1 - 1 node / 4 cores, 4 FPGAs)	LBL/NERSC	28	1,597,872,397	Convey Custom
26	IBM and ScaleMP/vSMP Foundation (128 cores)	LANL	29	1,551,460,000	Reference
27	SuperDragon-1 (32 nodes / 384 cores)	Institute of Computing Technology, Beijing, China	30 (Mini+)	1,454,110,000	Custom
28	PLX(32 nodes each with 2 Tesla M2070)	CINECA, Bologna, Italy	26	1,357,320,000	GPU Customized
29	cougarxmt (128 nodes)	PNL	29 (Mini)	1,223,375,650	Reference
30	GraphCREST-W12 (Intel Xeon CPU X5670 2.93GHz (2 sockets, 12 cores) / 1 node)	Chuo University, Tokyo Japan	15	1,191,232,322	Custom
31	graphstorm (128 nodes)	SNL	29 (Mini)	1,171,052,667	Reference
32	IBM and ScaleMP/vSMP Foundation (128 cores)	LANL	30	1,036,020,000	Reference
33	Jaguar (224,256 processors)	ORNL	30	1,010,500,000	Reference
34	Hyperion (64 nodes / 512 cores)	LLNL	31	1,004,005,127	Custom
35	Matterhorn (64 nodes)	CSCS	31	884,587,699	Reference
36	Minerva (3096 processors)	University of Warwick	26	839,444,000	Reference
37	GraphCREST-M48 (AMD Opteron Processor 6174 2.2 GHz (4 sockets, 48 cores) / 1 node)	Chuo University, Tokyo Japan	16	833,589,301	Custom
38	Erdos (64-MTA)	ORNL	29 (Mini)	701,767,082	Reference
39	Hyperion + FusionIO (64 nodes / 512 cores)	LLNL	36	609,481,208	Custom/NAND FLASH
40	ultraviolet (4 processors / 32 cores)	SNL	29	420,412,833	Custom
45	Owens (2 Intel Xeon CPU E5520 @ 2.27GHz with Hyper-Threading enabled)	The Ohio State University	26	266,125,887	Reference
41	Trestles (144 nodes / 4608 cores)	UCSD	36	242,623,828	Custom/NAND FLASH
42	Knot (64 cores / 8 processors)	UCSB	30	176,627,000	Custom/NAND FLASH
43	Kraken (6128 cores)	LLNL	31	104,644,000	Custom/NAND FLASH
44	Caribou (4 Intel Xeon X7560 processors, 2.27 GHz)	Link Analytics	27	78,559,283	Reference
46	Kraken (6128 cores / Fusion I/O)	LLNL	34	55,948,453	Custom/NAND FLASH
47	Leviathan + FusionIO (1 node / 40 cores)	LLNL	36	55,515,188	Custom/NAND FLASH
48	Neumann (32 cores)	UCSB	26	39,566,000	Custom
49	Gordon (7 nodes / 84 cores)	SDSC	29	29,994,123	Reference

Updated to reflect processing errors by the Graph 500 Steering Committee.

<sup>10</sup><http://www.graph500.org/nov2011.html>



Convey Computer Corporation  
1302 E. Collins Boulevard  
Richardson, Texas 75081  
Phone: 214.666.6024 Fax: 214.576.9848  
Toll Free: 866.338.1768  
[www.conveycomputer.com](http://www.conveycomputer.com)

CONV-11-023.4 © 2011 Convey Computer Corporation. Convey Computer, the Convey logo, HC-1 and Convey HC-1® are trademarks of Convey Computer Corporation in the U.S. and other countries. Printed in the U.S.A.