As data sets increase in size, extreme-scale computing is needed to organize and analyze massive data. This clearly affects computing integrity and cybersecurity. One of the research recommendations in the first 2015 ASCR Cybersecurity Workshop report [4] was: “Research and develop methods to collect extreme-scale data and knowledge, and develop and apply analytics in order to understand and improve scientific computing integrity and computer security”. Since this type of data is very different from those in traditional physics-based simulations, a significant effort is needed to develop the fundamental computational tools necessary for computer security and scientific computing integrity. Our goal is to enable scalable data analytics using existing and future DOE leadership-class facilities.

Supercomputing centers currently use specialized hardware platforms for data analysis and management. Examples include large shared-memory systems (e.g., PSC Blacklight and Sherlock) and/or systems with large-scale flash-based storage (e.g., TACC Wrangler). Our recent focus has been on scalable data analytics using supercomputers. We foresee a convergence in architectures in such a way that they will be equally friendly (or challenging) to both traditional scientific computations and data-intensive computations. With the large investments that organizations like DOE are making in supercomputers, it is important to leverage these investments for our data analytics needs as well. As most large data sets cannot be stored in the memory of a single compute node, there is a need for distributed-memory processing. In recent work, we show that end-to-end analytics on massive data sets can be performed using just a small fraction of a supercomputer’s resources. We analyze the largest publicly-available hyperlink graph (the 2012 Web Data Commons graph with 128 billion edges) using 256 compute nodes of the NCSA Blue Waters supercomputer [7].

We believe the following research areas will be important:

**Manycore and thread-scalable data analysis algorithms.** The concurrency on each compute node has grown dramatically with recent manycore architectures. In the recent past, we have developed novel graph algorithms for traditional multicore computers [6] and performance portable versions of these algorithms for GPU and Xeon Phi [8] hardware. Considering that these manycore architectures are the basis for the next generation supercomputers, developing new algorithms for analytics of interest that can perform well in these architectures is crucial.

**In-situ data analysis for scientific integrity and cybersecurity.** To ensure the integrity of results from extreme scientific simulations, we expect that lightweight application-specific models and analytics will be used. These learning-based analytics may solve classification, clustering, or inference problems for pattern and anomaly detection. These analytics may also be used for self-healing and recovery from hardware and software faults. Massively parallel and an in-situ
deployment of these analytics minimizes data movement costs and also circumvents any security issues involved in data sharing with third-party tools.

**Data decomposition/partitioning of highly irregular sparse data.** We assume data sets will be larger than what fits in shared memory, so we target distributed-memory extreme-scale computers. These may have tens of thousands of nodes, so how to partition the data and assign to nodes becomes important. Usually, data can be represented as a graph or a sparse matrix. The traditional approach in scientific computing is a 1D (row) partition. However, this incurs too much communication when there are nearly dense rows/columns. It has been shown that 2D data partitioning is much more scalable for scale-free and power-law graphs [9, 1]. An issue is that the cost of computing the data partitioning itself may be too high compared to the computations that follow. Recently, we developed a new 2D partitioning method [1] that can use any 1D partition and the cost is similar. Traditional multilevel graph or hypergraph partitioners are too slow and produce mediocre quality partitions. New research is needed into fast and high-quality partitioners for scale-free graphs. A good starting point might be label propagation (PULP) [5].

**Optimized numerical algorithms/software for irregular data.** Linear algebra is a valuable analysis tool, and solving linear systems, least-squares systems, or computing the SVD are core computational kernels. Much work has been done to develop algorithms and implement these in libraries such as PETSc and Trilinos [3]. However, many implicit assumptions have been made, for example that all vertices have roughly the same degree (rows have similar number of nonzeros), which is not true for many problems in data analytics. We therefore need to revisit the algorithms and the implementations in scientific software libraries to remove such implicit assumptions that become performance bottlenecks.

**Randomized numerical linear algebra and graph algorithms.** Randomization and sampling have proven important tools to handle large data sets. In numerical linear algebra, sketching based on sampling and random projections have led to efficient approximation algorithms for e.g., matrix multiplication, SVD, and least-squares problems. In graph algorithms, sampling can be used to examine only a fraction of the data (graph) and may lead to sublinear time algorithms. Such methods are important when the data is too big to be stored in memory or even on disk.

**Graph algorithms building blocks.** A topic of great current interest is the development of abstractions/concepts for graph algorithms that also can lead to good performance, known as graph algorithms building blocks (GABB). Several approaches have been proposed. Some vertex-edge based libraries such as MTGL and PBGL give high performance but are too low-level, while high-level interfaces often give poor performance. A recent trend is to use sparse linear algebra as the underlying approach to do graph algorithms, e.g. CombBLAS [2] and the emerging GraphBLAS. This approach is promising as a lot of investment in sparse matrix computations can be leveraged. However, not every graph algorithm can be efficiently reformulated this way. Also, this approach tends to give coarse-grained algorithms with high memory cost. We believe further research is needed to find better balance between high productivity (ease of use) and high performance.

**References**


