ExM: System support for extreme-scale, many-task applications

Ian Foster (PI), Ewing Lusk (PI), Ketan Maheshwari, Todd Munson, Michael Wilde (Lead PI), *Argonne*
Daniel S. Katz (PI), Justin Wozniak, Zhao Zhang, *University of Chicago*
Sameer Al-Kiswany, Matei Ripeanu (PI), Emalayan Vairavanathan, *University of British Columbia*

Problem: scaleup of many-task applications

Exascale computers will enable and demand new problem solving methods that *involve many concurrent and interacting tasks*. Methodologies such as rational design, uncertainty quantification, parameter estimation, and inverse modeling all have this many-task property. All will frequently have aggregate computing needs that require exascale computers. For example, proposed next-generation climate model ensemble studies will involve 1,000 or more runs, each requiring 10,000 cores for a week, to characterize model sensitivity to initial condition and parameter uncertainty.

Goal

The goal of the ExM project is to achieve the technical advances required to execute such many-task applications efficiently, reliably, and easily on petascale and exascale computers. In this way, we will open up extreme-scale computing to new problem solving methods and application classes.

Anticipated impact

The project will produce advances in computer science and software technology that enable the efficient and reliable use of exascale computers for new classes of applications. In this way, the project will both accelerate access to exascale computers by important existing applications and facilitate uptake of large-scale parallel computing by other application communities for which it is currently out of reach. The project will also produce students and postdocs expert in innovative system software for extreme-scale computers.

Project overview

Running many-task applications efficiently, reliably, and easily on extreme-scale computers is challenging. System software designed for today's mainstream single program multiple data (SPMD) computations is not necessarily a good match to the demands of many-task applications. To address these demands, the ExM project will design, develop, and evaluate two new system software components. The ExM data store will allow concurrent and asynchronous application tasks to communicate efficiently and reliably, both with each other and with persistent storage, by reading and writing data objects maintained in node-local storage, including memory, SSD, and local disk. The ExM task manager will allow for the rapid, data-aware, and efficient dispatch of many tasks to large exascale computing systems and for the fault-tolerant execution of those tasks. These components will be efficiently integrated with current and future extreme-scale system software and made available to developers via both a parallel scripting language and APIs.

Project scope and activities

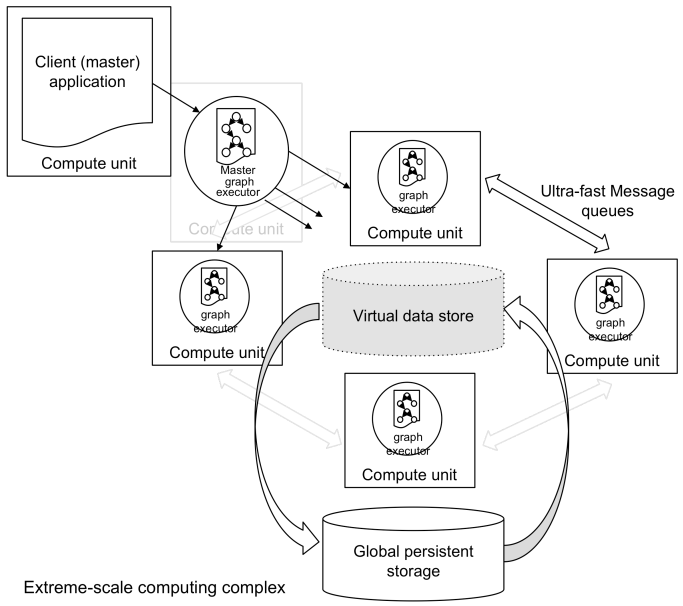
Design, build, and evaluate an *ExM data store* to provide efficient and reliable read and write operations from many tasks against both in-memory and on-disk storage systems, via both sequential and parallel I/O interfaces.

Design, build, and evaluate an *ExM task manager* to provide efficient and reliable management of large task graphs that can productively utilize exascale platforms.

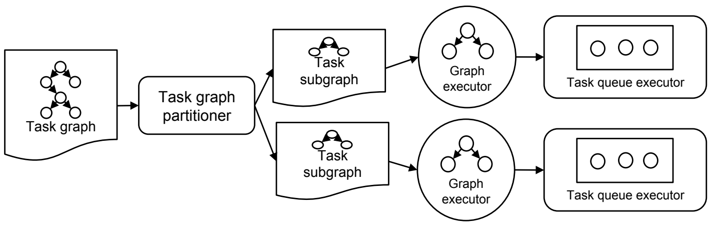
Integrate these components with high-level libraries and languages: C, Python, and Swift (as exemplars of distinct classes of high-level languages).

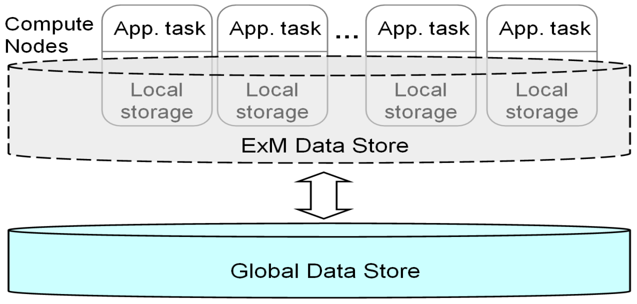
Evaluate the performance and usability of these system components, programming libraries, and systems at large scale by applying them to challenging DOE science applications.

ExM Architecture

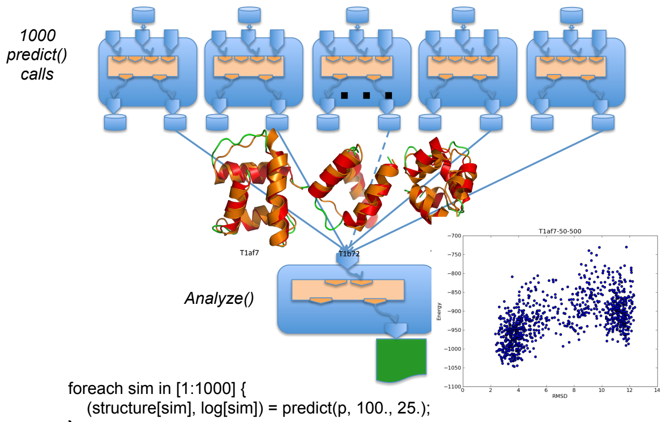


ExM distributed task management targets high utilization, low latency, and resiliency in the face of failing components and interconnects.

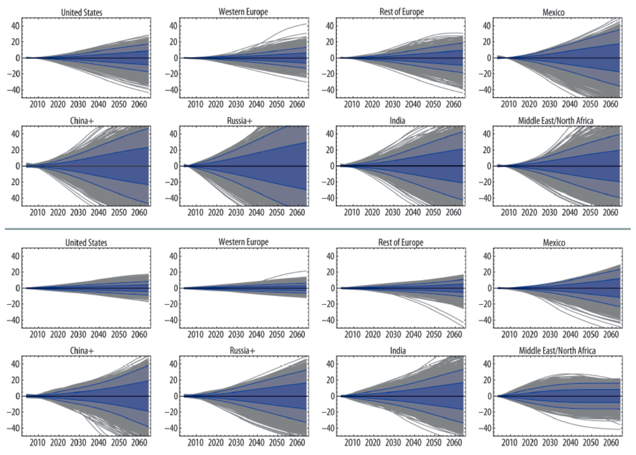




ExM complex-wide data storage – based on MosaStore – is embedded and distributed across nodes and RAM storage to provide a namespace and fast file data exchange.

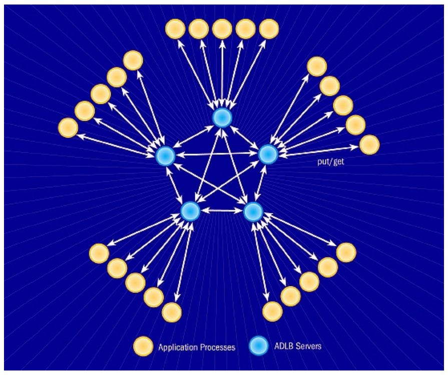


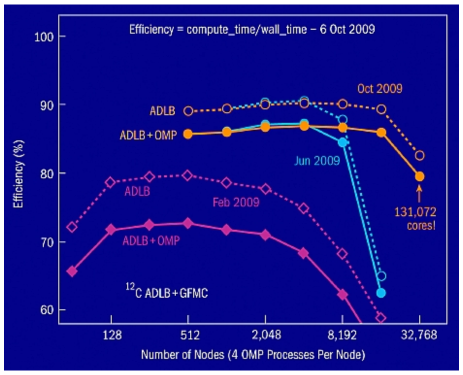
```
foreach sim in [1:1000] {
    (structure[sim], log[sim]) = predict(p, 100., 25.);
}
result = analyze(structure)
```



Simple *Swift* scripts (left) specify and execute 1,000 parallel tasks for protein structure prediction with Monte Carlo simulated annealing, and 5,000 models exploring uncertainty in consumer and industrial electricity usage (right).

ADLB and Swift provide initial starting points for ExM programming models.





Traditional Master/Worker has a scaling bottleneck of worker ranks trying to access the shared queue managed by the master. ADLB scales to over 100K cores by making a shared queue accessible to all ranks without need to go through an arbiter process.

For more information: **Contact: Michael Wilde, wilde@mcs.anl.gov**

Parallel Scripting for Applications at the Petascale and Beyond. Michael Wilde, Ian Foster, Kamil Iskra, Pete Beckman, Zhao Zhang, Allan Espinosa, Mihael Hategan, Ben Clifford, Ioan Raicu IEEE *Computer*, Vol. 42, 11. 2009.
Swift parallel scripting language: http://www.ci.uchicago.edu/swift
ADLB (Asynchronous dynamic load balancer): http://www.cs.mtsu.edu/~rbutler/adl*b*
MosaStore: http://netsyslab.ece.ubc.ca/wiki/index.php/MosaStore
ExM Project: http://www.mcs.anl.gov/exm (in progress)