***My greatest invention was creating an innovative peer-to-peer file-sharing protocol for the secure distribution of large sets of heterogeneous data over wide-area networks.* – Ian Kelley**

Volunteer computing is a popular and successful means of providing vast amounts of computational power to scientists for little or no direct cost. This is achieved through the creation of a publicly open system where private individuals (i.e., "volunteers") install a software program that takes advantage of their computer's processing power when it would otherwise be idle.

Launched in 1999, the SETI@Home project was instrumental in both the popularity of volunteer computing and the technical achievements that made it possible. SETI@Home enabled tens of thousands of non-scientists to contribute to science by installing a simple "screen saver" on their home computers. Behind the scenes, the program was doing much more than displaying an animation: it was analyzing vast amounts of radio telescope data, all part of SETI's search for extraterrestrial intelligence. Following SETI@Home's success, a generic version of its distributed computing software was created for wide-scale use by other scientific projects: the Berkeley Infrastructure for Open Network Computing (BOINC). BOINC provides a framework that allows any parallel processing application to easily leverage the power of the volunteer computing ecosystem. To date, >50 scientific projects have used BOINC to analyze their data, leveraging ~1m unique computers and creating a sustained 21.6 petaflops of computational power,[1] rivaling the fastest supercomputers in the world.[2]

BOINC has proven itself extremely successful at exploiting idle CPU and GPU cycles, yet it provides no sophisticated data management capabilities. Work units (i.e., tasks) in BOINC simply contain the *http* endpoint of an input file to process (see Figure 1), leaving project administrators with the responsibility of managing data and the resulting network loads. This is typically accomplished by deploying a series of globally distributed and high-performant mirror servers. Although this approach is viable, it has had the following repercussions on BOINC projects and volunteer computing adoption:

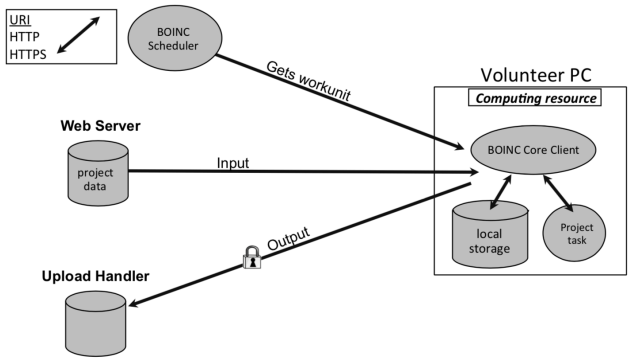

Figure 1: BOINC - Centralized Data Distribution

- data gets down-sampled, as projects reduce their data resolution to limit network and storage demands
- data-intensive workloads cannot take advantage of BOINC because their data distribution requirements are too large
- smaller projects and/or teams do not use BOINC because of the high network bandwidth requirements and data management costs
- transient jobs and workloads are typically unsuitable for BOINC, as they lack persistent infrastructure and dedicated support

In addition to these limitations, the centralized architecture of BOINC also creates a small number of failure points and potential bottlenecks, while failing to take advantage of plentiful client-side network bandwidth and storage capabilities. If these latent "volunteer" resources could be further exploited to include the storage and distribution of data sets, several of the limitations of the BOINC ecosystem would be mitigated, allowing for new data-intensive workloads to be supported while expanding BOINC's user community and increasing its impact.

Peer-to-Peer Architecture for Data Intensive Cycle Sharing

In 2005, I began research to develop a new data distribution system that leveraged peer-to-peer (P2P) technologies and decentralized data-sharing networks. The goal of these efforts was to support new distributed computing scenarios such as data-intensive workloads in systems like BOINC. By 2008, my research had resulted in

[1] BOINC project statistics [link]
[2] Top500 supercomputer list [link]

a "Peer-to-Peer Architecture for Data Intensive Cycle Sharing" (P2P-ADICS) that included a new file-sharing protocol, dubbed *attic*, and its reference implementation.[3]

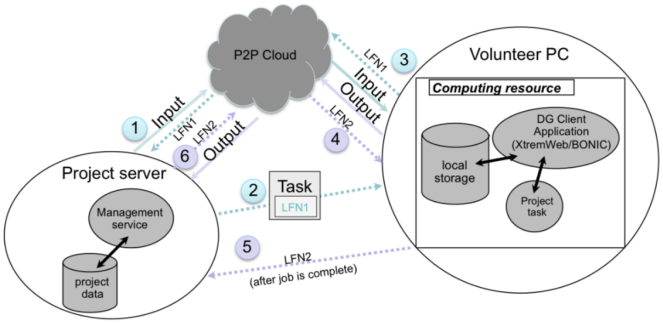

Figure 2: BOINC - Decentralized Data Distribution

In P2P-ADICS, rather than data being served by a centralized system as depicted in Figure 1, data was instead published to a P2P Cloud, which then served client requests.

Figure 2 outlines this process, with steps 1-3 showing the input data flow to clients, and steps 4-6 representing the flow of output data from clients back to project servers.

By offloading data distribution to the network, scalability and fault-tolerance are dramatically increased and peak-demand issues that result from network congestion can be marginalized. This can result in reduced total execution times for BOINC work units, as shown in Figure 3, which visualizes laboratory tests that compared a standalone BOINC server to 1, 3, and 9 peer-to-peer data sharing nodes under different load patterns.

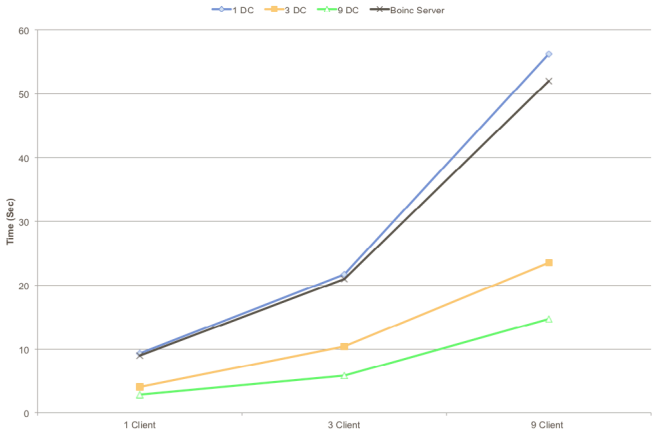

Figure 3: BOINC vs. 1, 3, 9 P2P data sharing nodes (DCs)

On the surface, the P2P-ADICS system is similar in nature to other peer-to-peer networks such as Napster, Gnuttella, BitTorrent and Skype. This is because even though each of these systems addresses unique use-cases for data management, they each share common challenges and thus, related solutions.

For example, any open and publicly available peer-to-peer network is by nature an unstable environment comprised of untrusted nodes. To mitigate these security concerns and to provide stability and scalability, similar design patterns, algorithms, and network topologies are leveraged.

One common pattern in highly-scalable P2P networks is to create a "super-peer" topology for message routing, which overcomes many of the limitations of a purely unstructured "flat" network. These "super peers" form a network overlay and act as metadata cashes that each keep track of a defined subset of the network. This topology facilitates fast searching and information retrieval, albeit with the cost of increased network complexity.

Beyond unique security and scientific application integration considerations, one of the key differentiators in the design of P2P-ADICS and the *attic* protocol was that they needed to operate in an environment that tolerates "free loading" of the data network, which is where some participants consume data without also sharing. This trait is atypical of P2P systems, which usually institute "tit for tat" policies to ensure fair play and parity between the aggregate throughput of data providers and data consumers (i.e., uploaders and downloaders).

For P2P-ADICS, it was not necessary to enforce that consumers of data also act as data providers.. This was due to the volunteer nature of the system that allowed data consumers to contribute in other ways, such as by donating spare CPU cycles. Designing a system that functioned in this manner was key, since customer (i.e., volunteer) trust

[3] Ian Kelley, Ph.D. dissertation, Data management in dynamic distributed computing environments. [link]

could easily be broken if their networks became congested and volunteers began to suffer adverse consequences as a result of their participation.

P2P-ADICS was developed by taking concepts from several different P2P systems, e.g., the double hashing and file-swarming techniques of BitTorrent, coupled with a variation of the brokered super-peer topologies seen in late Napster implementations. The ultimate goal was to minimize network entities to core functions, limit message traffic and upkeep due to churn, provide a central authority that can easily be governed, secured, and managed, while enabling a network paradigm that supports segmenting the network into *providers* and *clients* to facilitate volunteer computing's unique contribution/attribution needs.

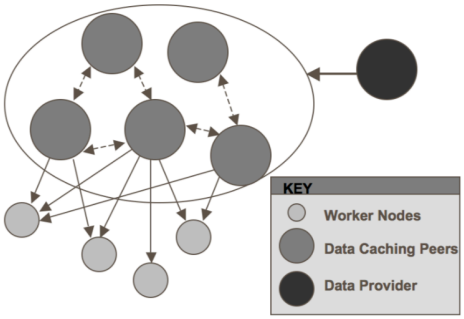

Figure 4: P2P-ADICS Network Topology

Figure 4 shows the final network architecture of P2P-ADICS. In it, three core entities are depicted: (1) worker nodes, which download and process data, (2) data caching peers, which replicate data on the network and form a special overlay network, and (3) data providers, which publish data to the network.

Not depicted in the diagram are data lookup servers, which are responsible for indexing metadata to keep track of which data caching peers have replicas (or partial replicas) of a requested file.

The *attic* protocol was developed to facilitate network actions and entity interactions in the P2P-ADICS network. *attic* was based on *http*, which provided a straightforward integration with existing systems and support for legacy applications. Services were exposed in a RESTful manner and a common vocabulary and taxonomy was defined to provide extensible item metadata and extensibility.

EDGeS, EDGI and Beyond

As a direct result of the innovative research undertaken on P2P-ADICS, I successfully completed my Ph.D. and the research project that funded it.

Beyond that, the research and software foundation P2P-ADICS provided allowed me to lead Cardiff University's efforts on several joint grant proposals to the European Union. After they were successfully funded, I hired and led researchers to further develop the P2P-ADICS software, while expanding its impact to use-cases in new domains.

In the *Enabling Desktop Grids for e-Science* (EDGeS) project,[4] a $4m effort that lasted from 2008 – 2010, P2P-ADICS was the key foundation for all work undertaken in the "Data Access" joint research activity. During this time, the *attic* protocol evolved, protocol handlers and adapters were built for BOINC and other software systems, and the system was deployed to numerous locations throughout Europe.

In the *European Desktop Grid Initiative* (EDGI) project,[5] a $3m effort that lasted from 2010 – 2012, P2P-ADICS was the key software system for data migration and hosting between high-performance computing (HPC) infrastructures (e.g., supercomputers) and volunteer computing environments (e.g., BOINC). P2P-ADICS acted as a glue layer between these two systems that facilitated the transfer of highly parallel computational workloads from (extremely expensive) HPC systems (e.g., EGEE) to cheaper processing alternatives (e.g., volunteer computing).

In addition, P2P-ADICS was used as the foundation for several other distributed computing research projects at Cardiff University. In these cases, P2P-ADICS was used as a simulation testbed and/or the *attic* protocol was extended to support new features such as sophisticated quality of service (QoS).

[4] https://cordis.europa.eu/project/rcn/86420_en.html
[5] https://cordis.europa.eu/project/rcn/95201_en.html