Improving Data Retrieval Rates Using Remote Data Servers

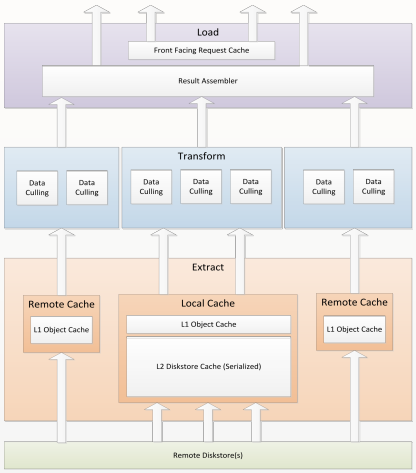
Ted D'Ottavio, Bartosz Frak, Seth Nemesure and John Morris, BNL, Upton, NY, USA*

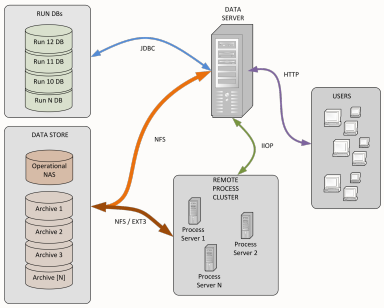
ABSTRACT

The power and scope of modern Control Systems has led to an increased amount of data being collected and stored, including data collected at high (kHz) frequencies. One consequence is that users now routinely make data requests that can cause gigabytes of data to be read and displayed. Given that a users patience can be measured in seconds, this can be quite a technical challenge. This paper explores one possible solution to this problem - the creation of remote data servers whose performance is optimized to handle context-sensitive data requests. Methods for increasing data delivery performance include the use of high speed network connections between the stored data and the data servers, smart caching of frequently used data, and the culling of data delivered as determined by the context of the data request. This paper describes decisions made when constructing these servers and compares data retrieval performance by clients that use or do not use an intermediate data server.

ARCHITECTURE

Client applications issue requests over a RESTful based API to the Data Server. The server retrieves a file list from one or more run databases. These individual files are read from a network store(s), processed (culled) and cached in parallel in the same VM or on one or more satellite server. The complete or partial results are transformed to XML or JSON and shipped back the client. In the simplest case, which involves complete reads, the requests are completely stateless, however minimal state information has to be maintained for clients, which ask for partial results.

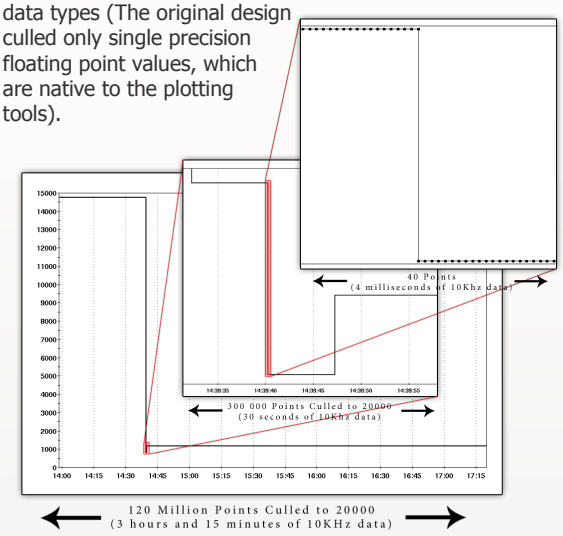




Logger data, in either its processed and raw forms exists and can be retrieved from one of four locations. At the highest level, the Front Facing Request Cache (L0) returns processed results for requests it had already seen. Every request, which falls through the L0 cache is dispatched through a local or remote object cache (L1), which either returns immediately with the raw data or issues a new read request to a remote disk store. In addition to the L1 object cache, local server instance has a transparent access to a SSD based cache (L2), which holds serialized, memory-store evicted instance fragments. At a level above, regardless of its spatial location, all data fragments have to go through a transform phase, which currently simply cuts down on their density, by subjecting them to the culling algorithm.

DATA CULLING

Virtually every request, which passes through the Data Server is subjected to the culling algorithm. This algorithm was designed to cut down high volume, time domain data to more manageable, lower rate datasets, which look virtually the same on the users' monitors as their original, high frequency counterparts. This algorithm was originally implemented in our C++ plotting library and used with great success in the LogView application. A Java version, which was incorporated in the Data Server, extends the original design by parallelizing the processing pipeline. This implementation allows the server to process up to a billion points per second and adds the ability to cull all numerical data types (The original design culled only single precision floating point values, which are native to the plotting tools).



RESULTS

	Throughput (MB/sec)	Speedup
Client to Remote Disk-store	5.4	
Client through Data Server to Remote Disk-store	146	27x
Client through Data Server to SSD Cache	245	45x
Client through Data Server to RAM	968	180x

Test suite used to compile the following result table included a combination of both high and low frequency data reads over varying periods of time. Throughput was measured only for requested data, i.e. additional information stored in the files, but not explicitly asked for was discarded and not used in the rate calculations. The final throughput was calculated by dividing the data volume by the sum of the extract, transform, load and client processing times. Individual tests have been repeated multiple times to minimize the any rate variances.

BROOKHAVEN NATIONAL LABORATORY

*Work supported by Brookhaven Science Associates, LLC under contract no. DE-AC02-98CH10886 with the U.S. Department of Energy