Dremel supports interactive analysis of very large datasets over shared clusters of commodity machines. It is capable of operating on in situ nested data. In situ refers to the ability to access data ‘in place’,e.g.. in a distributed file system or another storage layer[1]. Dremel can execute many queries over such data that would ordinarily require a sequence of MapReduce jobs, but at a fraction of execution time. Dremel is not intended as a replacement for MapReduce and is often used in conjunction with it to analyze outputs of MapReduce pipelines or rapidly prototype larger computations. A nested data model underlies most of structured data processing at Google and reportedly[3] at other major web companies.

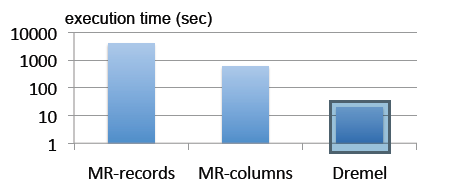
The Dremel’s data model is based on strongly-typed nested records. Its abstract syntax is given by:

[1]

where is an atomic type or record type. Atomic types in dom comprise integers, floating-point numbers, stirngs etc. Records consist of one or multiple fields. Field I in a record has a name and an optional multiplicity label. Repeated fields(\*) may occur multiple times in a record.

Each column is stored as a set of block. Each block contains the repetition and definition levels and compressed field values.

Dremel’s query language is based on SQL and is designed to be efficiently implementable on columnar nested storage.



We illustrate a MapReduce and Dremel execution on columnar vs record-oriented data.

The MapReduce framework was designed to address the challenges of large-scale computing in the context of long-running batches jobs. Like MapReduce, Dremel provides fault tolerant capabilities. Our columnar representation of nested data builds on ideas : separation of structure from content and transposed representation.

Dremel, a distributed system for interactive analisis of large datasets, complements the MapReduce paradigm. It performs on trillion-record, multi-terabyte and has its special storage format, query language, and execution.

The future main challenges will be formal algebratic specification, joins, and extensibility mechanisms.

Dynamo is a highly available and scalable distributed data store built for Amazon’s platform. Dynamo is used to manage the state of services that have very high reliability requirements and need tight control over the tradeoffs between availability, consistency, cost-effectiveness and performace. Dynamo uses a synthesis of well known techniques to achieve scalability and availability: Data is partitioned and replicated using consistent hashing, and consistency is facilitated[4]. Dynamo forces us to push the complexity of conflict resolution to reads in order to ensure writes are never rejected.

In Dynamo, each storage node has three main software components: request coordination, membership and failure detection, and a local persistence engine.

By using Dynamo, applications hava received successful responses for 99.9995% of its requests and no data loss event has occurred to date.

Moreover, the primary advantage of Dynamo is that it provides the necessary knobs using the three parameters of (N,R,W) to tune their instance based on their needs.[2]

Reference

[1] Dremel: Interactive Analysis of WebScale Datasets

[2] Dynamo: Amazon’s Highly Available Key-value Store

[3] Protocol Buffers: Developer Guide. Available at

<http://code.google.com/apis/protocolbuffers/docs/overview.html>.

[4] Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., and Lewin, D. 1997. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In Proceedings of the Twenty-Ninth Annual ACM Symposium on theory of Computing (El Paso, Texas, United States, May 04 - 06, 1997). STOC '97. ACM Press, New York, NY, 654-663.