Programming Models and Workloads for WSC

- The most popular framework for batch processing in a WSC is Map-Reduce [Dean and Ghemawat 2008] and its open-source twin Hadoop. Facebook runs Hadoop on 2000 batch-processing servers of the 60,000 servers it is estimated to have in 2011.

- Map first applies a programmer-supplied function to each logical input record. Map runs on thousands of computers to produce an intermediate result of key-value pairs. Reduce collects the output of those distributed tasks and collapses them using another programmer-defined function. With appropriate software support, both are highly parallel yet easy to understand and to use. Within 30 minutes, a novice programmer can run a MapReduce task on thousands of computers.

- For example, one MapReduce program calculates the number of occurrences of every English word in a large collection of documents. Below is a simplified version of that program, which shows just the inner loop and assumes just one occurrence of all English words found in a document.

```
map(String key, String value):
// key: document name
// value: document contents for each word w in value:
EmitIntermediate(w, "1");        // Produce list of all words

reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int result=0;
for each v in values:
result += ParseInt(v);      // get integer from key-value pair
Emit(AsString(result));
```

- The function EmitIntermediate used in the Map function emits each word in the document and the value one. Then the Reduce function sums all the values per word for each document using Parselnt() to get the number of occurrences per word in all documents. The MapReduce runtime environment schedules map tasks and reduce task to the nodes of a WSC.

- MapReduce can be thought of as a generalization of the single-instruction, multiple-data (SIMD) operation except that you pass a function to be applied to the data-that is followed by a function that is used in a reduction of the output from the Map task. Because reductions are commonplace even in SIMD programs, SIMD hardware often offers special operations for them.

- To accommodate variability in performance from thousands of computers, the MapReduce scheduler assigns new tasks based on how quickly nodes complete prior tasks. Obviously, a single slow task can hold up completion of a large MapReduce job. In a WSC, the solution to slow tasks is to provide software mechanisms to cope with such variability that is inherent at this scale. This approach is in sharp contrast to the solution for a server in a conventional datacenter, where traditionally slow tasks mean hardware is broken and needs to be replaced or that server software needs tuning and rewriting.

- Programming frameworks such as MapReduce for batch processing and externally facing SaaS such as search rely upon internal software services for their success. For example, MapReduce relies on the Google File System (GFS) to supply files to any computer, so that MapReduce tasks can be scheduled anywhere.

- WSC storage software often uses relaxed consistency rather than following all the ACID (***atomicity, consistency, isolation, and durability***) requirements of conventional database systems. The insight is

that it's important for multiple replicas of data to agree eventually, but for most applications they need not be in agreement at all times.