Introduction to Distributed Systems

Prof. Carl Hauser

Cpt. S 464/564 Lecture

Auxiliary Material (not from text)

August 24, 2004

Administrative Items

- Handouts today
  - Syllabus
  - Student Survey (Last page of Syllabus)
- Dealing with WHETS

Outline of Topics

1. **Introduction**
2. Comparison of Distributed and Parallel Computing
3. Example Local vs. Remote Procedure Call

Introduction

- A distributed system is “one in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing”
  - Very broad definition
  - Lots of examples
  - Lots of kinds
- Abbreviations
  - “Distributed System” by “DS”,
  - “Distributed Computing” is “DC”
- “You know you have one when the crash of a computer you’ve never heard of stops you from getting any work done.” Leslie Lamport
- Examples of DSs:
  -
  -
  -

Advantages of Distributed Systems

- Share resources (key)
- Share devices
- Better hardware cost/performance than supercomputers, multiprocessors
- Allows access from many remote users using their simple PCs
- Allows for incremental growth (if done right)
- Increases reliability and availability (if done right)
- Some applications and services are inherently distributed
- Can spread the load of a given service much more easily
- Can potentially increase security (!!!???)

Consequences of Distributed Systems

- Concurrency
  - Concurrent use of low-level resources: processing, storage (memory+disk), communications
  - Mutual exclusion and other synchronization required
  - Access to resources for a given user often best-effort
- No global clock
  - Cannot often know the exact ordering of events: which happened first
- Independent failures
  - No longer “all or none” failures for your program!
  - Some computers still running, while others failed or partitioned
  - Failure of a component you are using may not be a clean failure
- Security implications
  - More points of attack
  - Designs may be more susceptible to attack (because they specifically allow “remote access”)

Outline of Topics

1. Introduction
2. **Comparison of Distributed and Parallel Computing**
3. Example Local vs. Remote Procedure Call

Comparison: DC and Parallel Computing

- (Note: material from: Claudia Leopold, *Parallel and Distributed Computing: A Survey of Models, Paradigms, and Approaches*, John Wiley and Sons, 2001)

- Common characteristics
  - Multiple processors are used
  - Processors interconnected by some network
  - Multiple computational activities (processes) are in progress at the same time and cooperate with each other
- Some consider parallel computing a subfield of DC!
  - I disagree – they are very different…..
- Parallel computing splits an application up into tasks that are executed at the same time, whereas distributed computing splits an application up into tasks that are executed at different locations using different resources.

Differences: DC and Parallel Computing

- Parallel Computing puts emphasis on the following:
  - An application is split into subtasks that are solved simultaneously, often in a “tightly coupled” manner
  - One application is considered at a time, with the goal of speeding up the processing of that single application
  - Programs are generally run on homogeneous architectures, which typically have shared memory
  - Fault tolerance and security are not generally considered

Differences: DC and Parallel Computing (cont.)

- Distributed Computing puts emphasis on the following:
  - Computation uses multiple resources physically separated: processors, memory, disk, databases
  - Multiple applications run at a time for many users
  - Heterogenous systems, open and dynamic
  - No shared memory, at least not in hardware
  - Fault tolerance and security must be dealt with (in some manner)
  - Sometimes the emphasis is on hiding system internals in a manner that the distributed system looks like a single large machine. Feature called a *single system image*, used in *cluster computing.*

Convergence of DC and Parallel Computing

- Architectures approaching each other
  - Fast network technologies allow cluster computing
  - Parallel machines increasingly used as servers in a DS
- Parallelism and distribution are closely related
  - Main differences in distribution: delay and partial failures
- Some joint meetings of parallel and distributed researchers

Outline of Topics

1. Introduction
2. Comparison of Distributed and Parallel Computing
3. **Example Local vs. Remote Procedure Call**

Example Local Call

Caller:

```
// declare and init stuff
x = new int [100];
Y = new util;
Flag = y.sort(x, 100);
```

Callee:

```
// declare and init stuff
Int util:sort(int [] a, int max) {
      // implementation of sort
    return status;
}
```



- Potential assumptions:
  - Procedure call conventions between caller (“client”) and callee
  - In same address space (on same computer)
  - In same programming language (usually)
  - Written by same programmer (often, not always)
  - Can transfer data and control quickly, effectively in zero time
  - Both fail, or neither do (for the most part)
- None of these assumptions are true in a distributed system!

Example Remote Call

Caller:

```
// declare and init stuff
x = new int [100];
Y = new util.bind();
Flag = y.sort(x, 100);
…
```

```
// "proxy" or "stub"
// generated by middleware
int util:sort(int [] a, int max){
  // put a[], max into struct
  // send message with struct
  // receive message w/ struct
  // copy from struct to a[],
  // status
    return status;
}
```

Callee:

```
// declare and init stuff
int util_impl:sort(int [] a,
                   int max)        {
      // implementation of sort
    return status;
}
```

```
// "skeleton" generated
// by middleware compiler
  …
   // receive message with struct
   // copy from struct to a[], max
   flag = z.sort(a, max)
  // copy a[], flag into struct
  // send message with struct
  …
```

Many Assumptions do not Hold!

- Not a local procedure call, so need more help
- Not in same programming language (can't assume this)
- Not written by same programmer
- Not always in the same administrative domain
- Latency for transfer of control and data can be large and, worse, unpredictable
- Partial failures
- Membership of the system (the computers in its collection) can change
- Unreliable or insecure communication

Reading for Thursday

Chapter 1, V & R