COSC2737
IT Infrastructure and Security

Apache Multiprocessor Modules

Section 4.4

Overview

ITIS

- Apache Web Server Processing Architecture
- Processes abd Threads
- Process and Evebt Driven Architecture
- Multi-Processing Nodules
  - Pre-fork
  - Worker
  - Event
  - Winnt

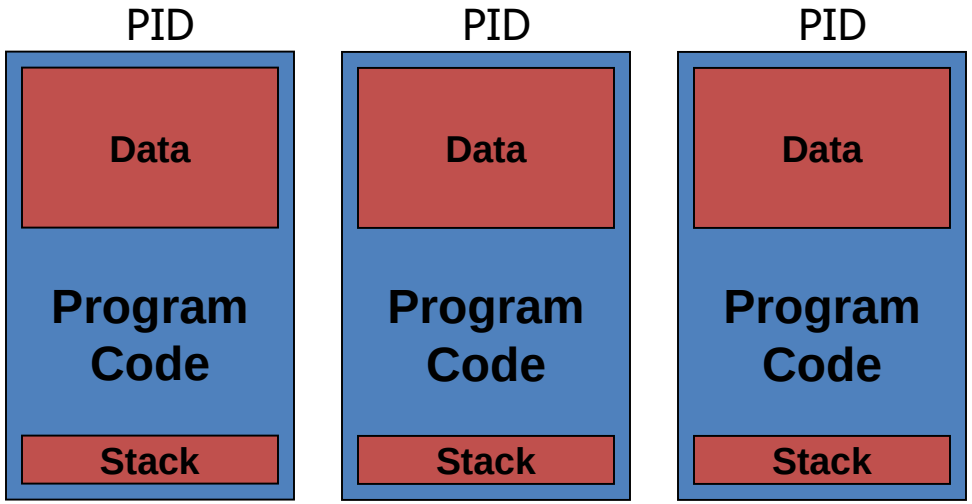
Web Server Processing Architecture

- A server will almost certainly have to handle more than one request at a time.
- When a request arrives, the server needs to use system resources: the processor, the disk, memory and network connections. Multiple requests mean the server has to have some way of sharing these resources between requests.
- There are two approaches:
  - Process-driven architecture
  - Event-driven architecture

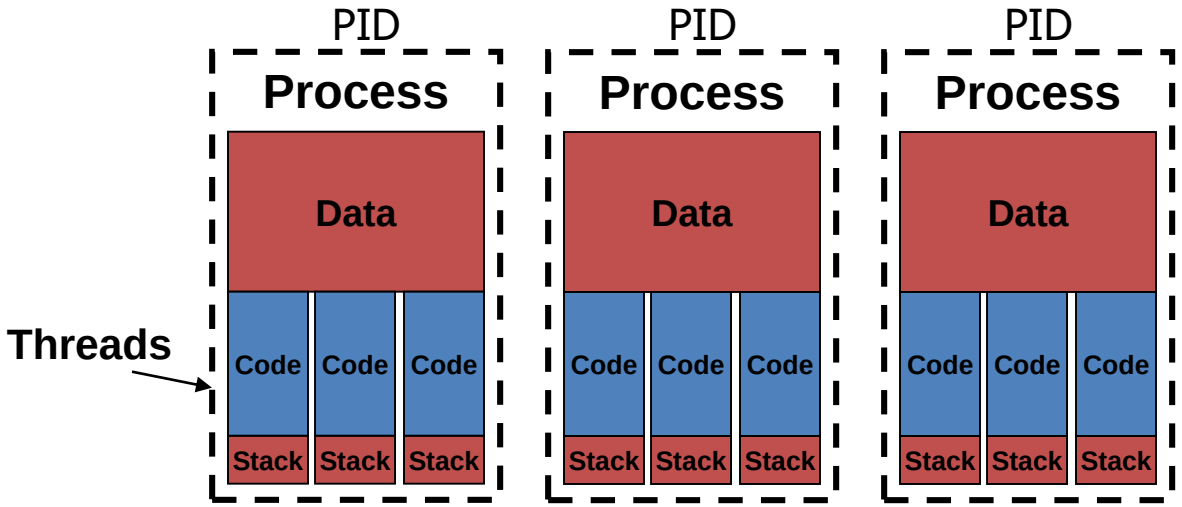
Processes and Threads

ITIS

- A *process* is a program in the process of executing, with all its supporting infrastructure.
  - It is independent
  - It has its own process ID (PID)



- Threads are lightweight strands of execution within a process.
  - A thread is part of a program and shares aspects of the program, like global variables, with other threads.
  - All processes have at least one thread, but may have more than one thread.



Process Driven Architecture

ITIS

- In a process-driven architecture, a new process is created for each request.
  - Usually, a single “master” process listens for requests and forks a separate process to deal with the request.

- Advantages
  - The process-driven architecture is a simple approach that requires simple software to implement.

- Disadvantages
  - The overhead of spawning and killing new processes is relatively high compared to the effort required to deal with HTTP requests.
  - This is why tuning a server such as Apache for optimum performance is so important.

Event Driven Architecture

ITIS

- In event-driven architecture, also known as multi-threading, a single process alternates between requests. A small amount of work is done on each request in turn.

- Advantages
  - Event-driven servers are fast and efficient. There is no overhead of starting/stopping/running multiple processes. A single process can also efficiently share data between requests.

- Disadvantages
  - A web server request can generally be subdivided into small amounts of work (assessing the request, finding the resource, returning data).
  - However, there can be significant delays at some of these steps.
  - For example, the web server has no control over the length of time a script will run.
  - Quite complex software is required to work around these issues.

Apache Multi Processing Modules

ITIS

- MPMs are used to tailor Apache to specific platforms.
  - A “orefork” MPM is included in Apache during compilation, which resembles the original Apache.

- The most important responsibilities of MPMs are:
  - setting up sockets and binding ports
  - starting processes and threads
  - managing communication between the parent and child processes
  - monitor child processes
  - switch user and group

Apache Multi Processing Modules

ITIS

- There are currently two major MPMs available for unix:
  - prefork (functions like the original Apache 1.3)
  - worker ('flagship' of Apache 2)
  - event (version of worker supporting more threads)
  - perchild, leader, threadpool (experimental MPMs)
- …and a number of specialised MPMs for different OSes
  - winnt, beos, netware, os/2
- We will look at prefork, worker, event and winnt in detail

Default MPMs

ITIS

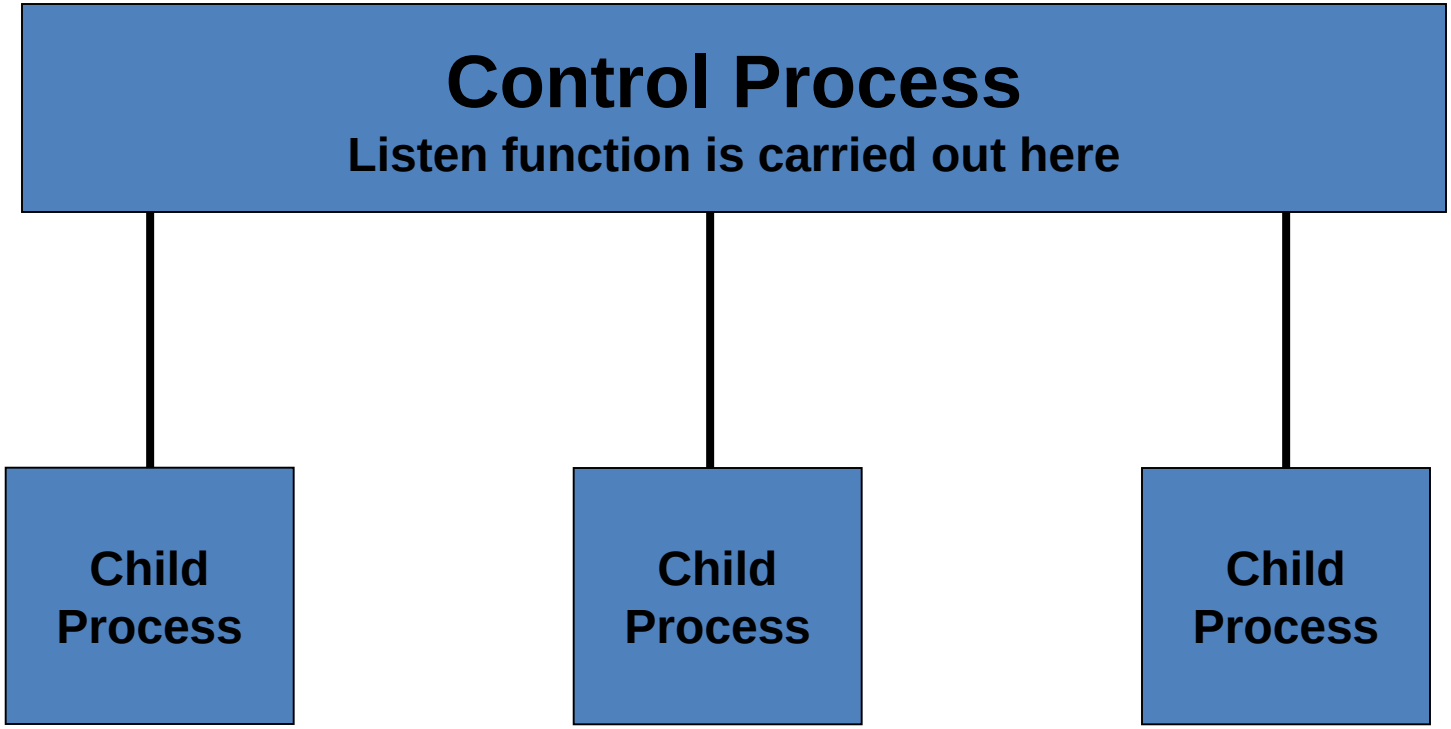
- In the case of Windows, the default MPM is ***winnt***.
- In the case of Unix or MacOS, the decision as to which MPM is installed is based on two questions:
  1. Does the system support threads?
  2. Does the system support thread-safe polling (Specifically, the `ekqueue()` and `epoll()` functions)?
- The default MPM used depends on the above questions.
  - If the answer to both questions is 'yes', the default MPM is ***event***.
  - If The answer to #1 is 'yes', but to #2 is 'no', the default is ***worker***.
  - If the answer to both questions is 'no', then the default is ***prefork***.
- In practical terms, this means that the default will almost always be ***event***, as all modern operating systems support these two features.

Prefork MPM

- The functionality of the prefork MPM is identical to that of Apache 1.3.x. That is, it is based on processes.
  - A parent process creates and controls a number of child processes.
  - Each child process listens to a socket for a request and responds to that single request.
- The prefork MPM is considered robust
  - if a fault occurs with one of the processes, then only a single request is lost.
- However using processes in this way is not efficient, and the prefork MPM suffers from scalability problems as it requires a large amount of RAM per request.

Prefork MPM

ITIS



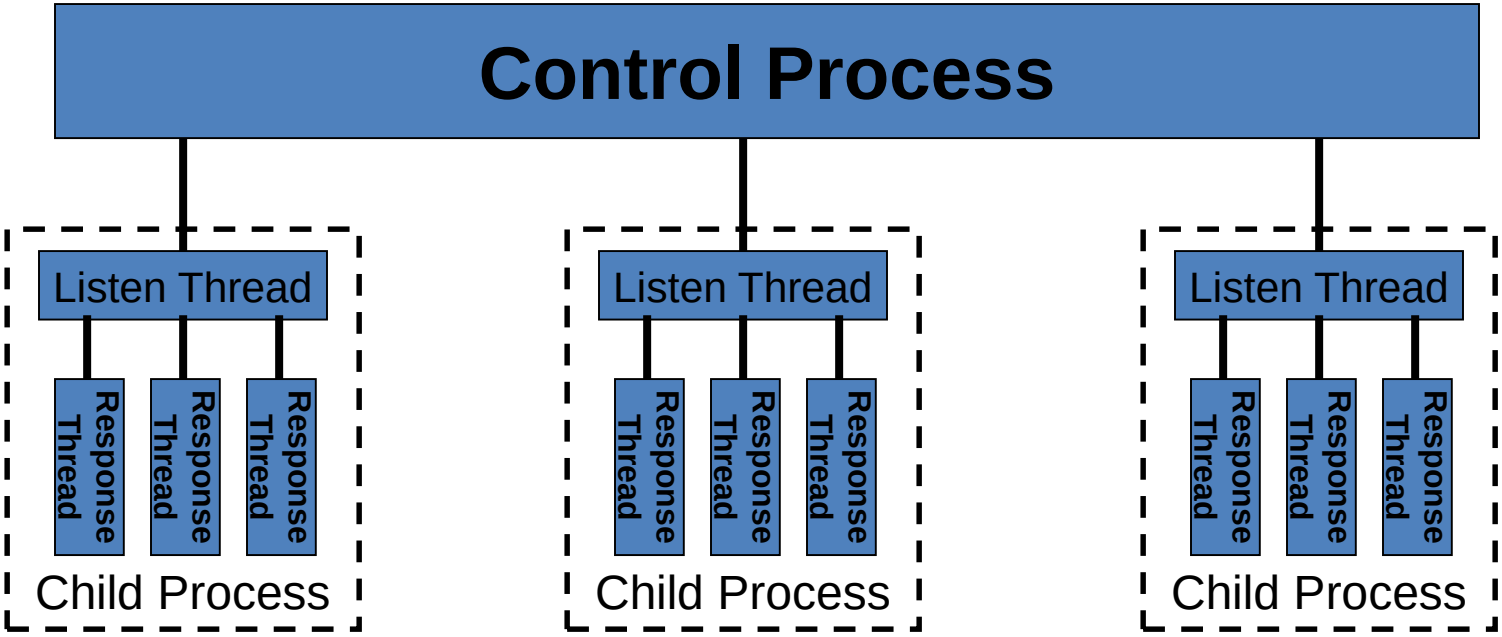
Worker MPM

- The worker MPM is a hybrid of processes and threads.
  - Like prefork a parent controls a number of child processes.
  - However each child process creates a set number of threads.
    - One thread listens for connections, and places them in a list.
    - The other threads read that list and respond to the requests.

- So the worker MPM uses much less RAM per request and therefore has much better scalability.
  - However it is less stable, as the loss of one process means the loss of multiple requests.

Worker MPM

ITIS



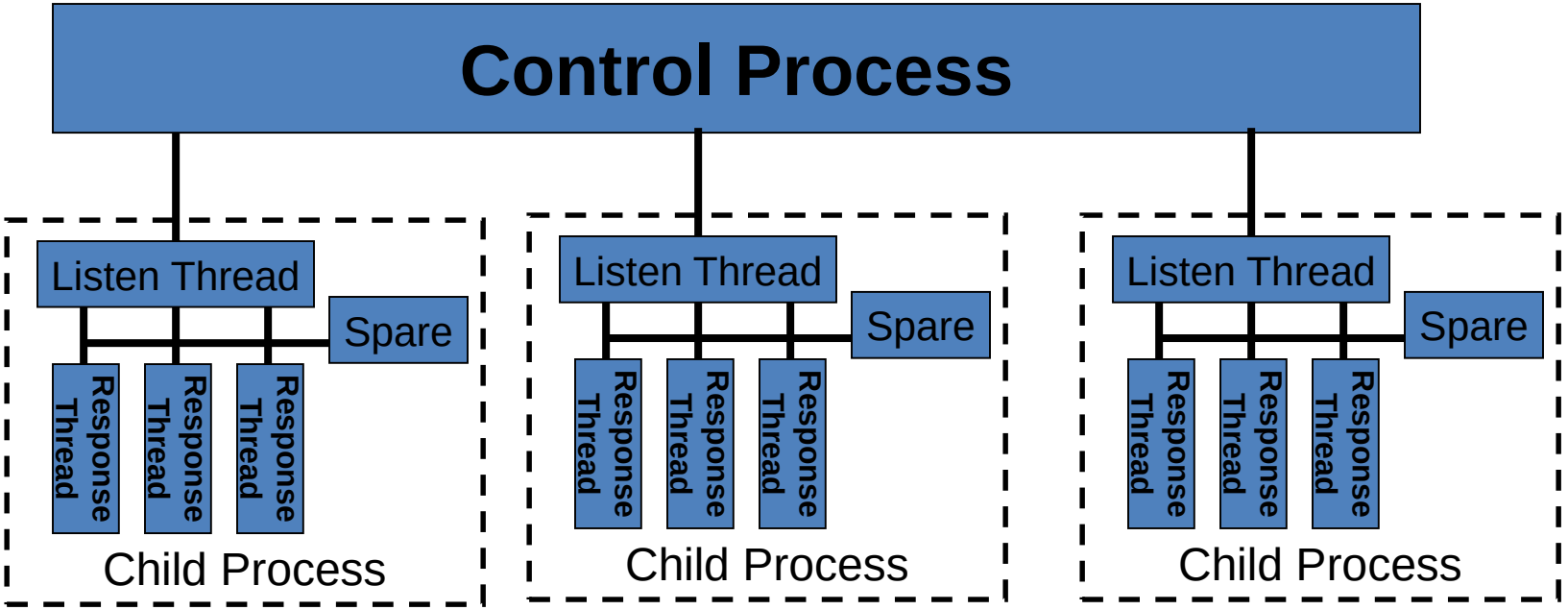
Event MPM

ITIS

- The event MPM is a hybrid of the worker MPM
  - It addresses the 'keep-alive' problem in HTTP
    - After an initial response, 'keep-alive's are used to keep the connection open, which can be very beneficial for net utilisation,
    - But this uses up an entire thread which idles and is a waster of memory
  - Instead, we separate the listening threads and pool the sockets in a 'keep-alive' state.
    - So the 'Listen' thread passes the initial message to a response thread, and when they are set to wait, they pass to the 'Spare' thread, until some activity causes the newly arrived message to be passed back to a thread (not necessarily the same as before).
    - This way the response threads are kept free to response
  - This is important for handling AJAX clients

Event MPM

ITIS



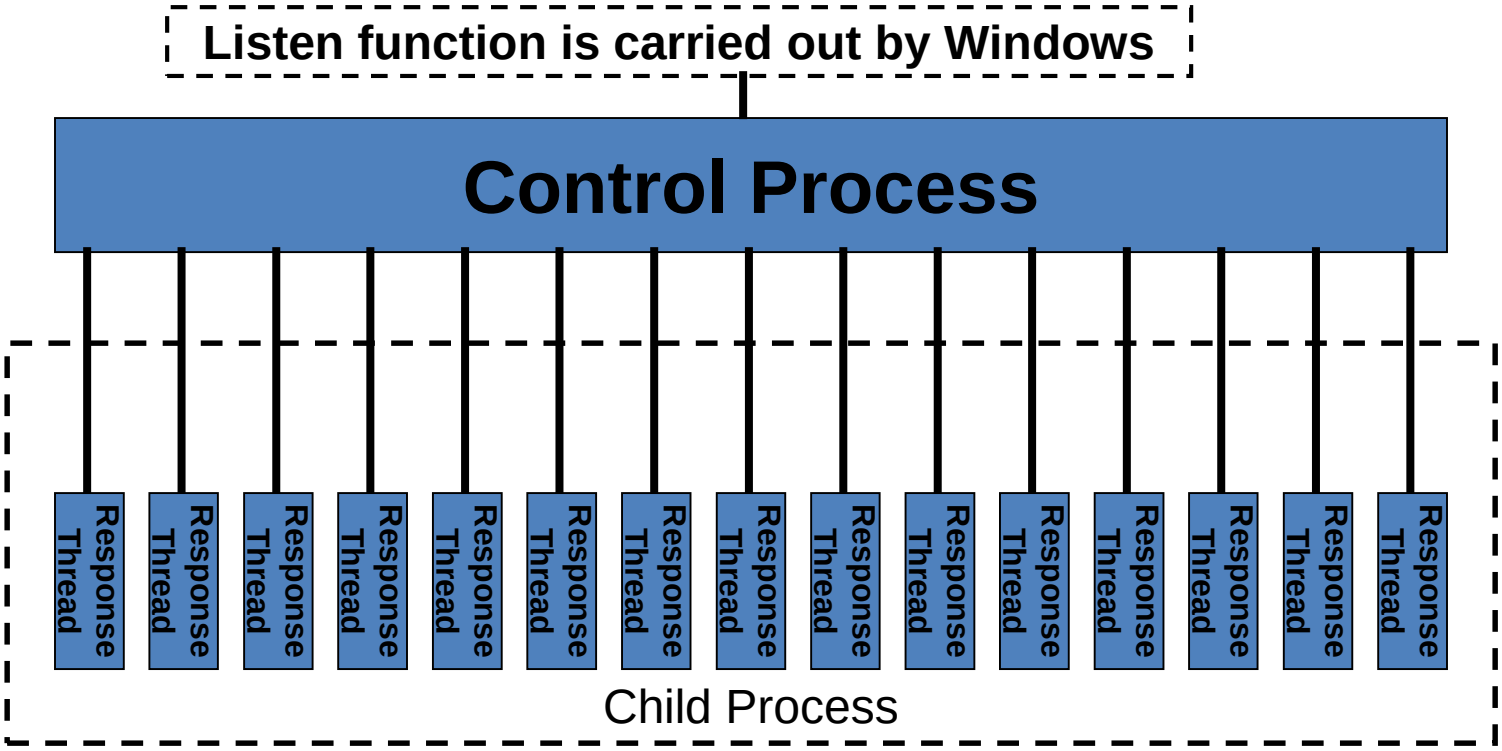
Winnt MPM

- The winnt MPM does not support multiple processes.
  - It is an event-driven only architecture, designed to take advantage of the features of Windows.

- The winnt MPM creates a single parent process and a single child process.
  - The child process has a set number of threads which respond to incoming requests.

Winnt MPM

ITIS



Effect of MPMs

- The use of Multi-Processor Modules has had a major effect in increasing the life of the Apache server.
  - Version 1.3 was starting to look a bit dated even in 2000, and the code architecture really showed how it got its name: "*a patchy server*".
  - Version 2.0 and its use of MPMs changed all that and made it one of the most flexible pieces of server software in common use.
    - the only downside is that changing modules requires recompilation
- studying the source code architecture is an excellent way of learning how to set up real-time systems.

Summary

ITIS

- Apache Web Server Processing Architecture
- Processes abd Threads
- Process and Evebt Driven Architecture
- Multi-Processing Nodules
  - Pre-fork
  - Worker
  - Event
  - Winnt