Scheduling (Win 2K)

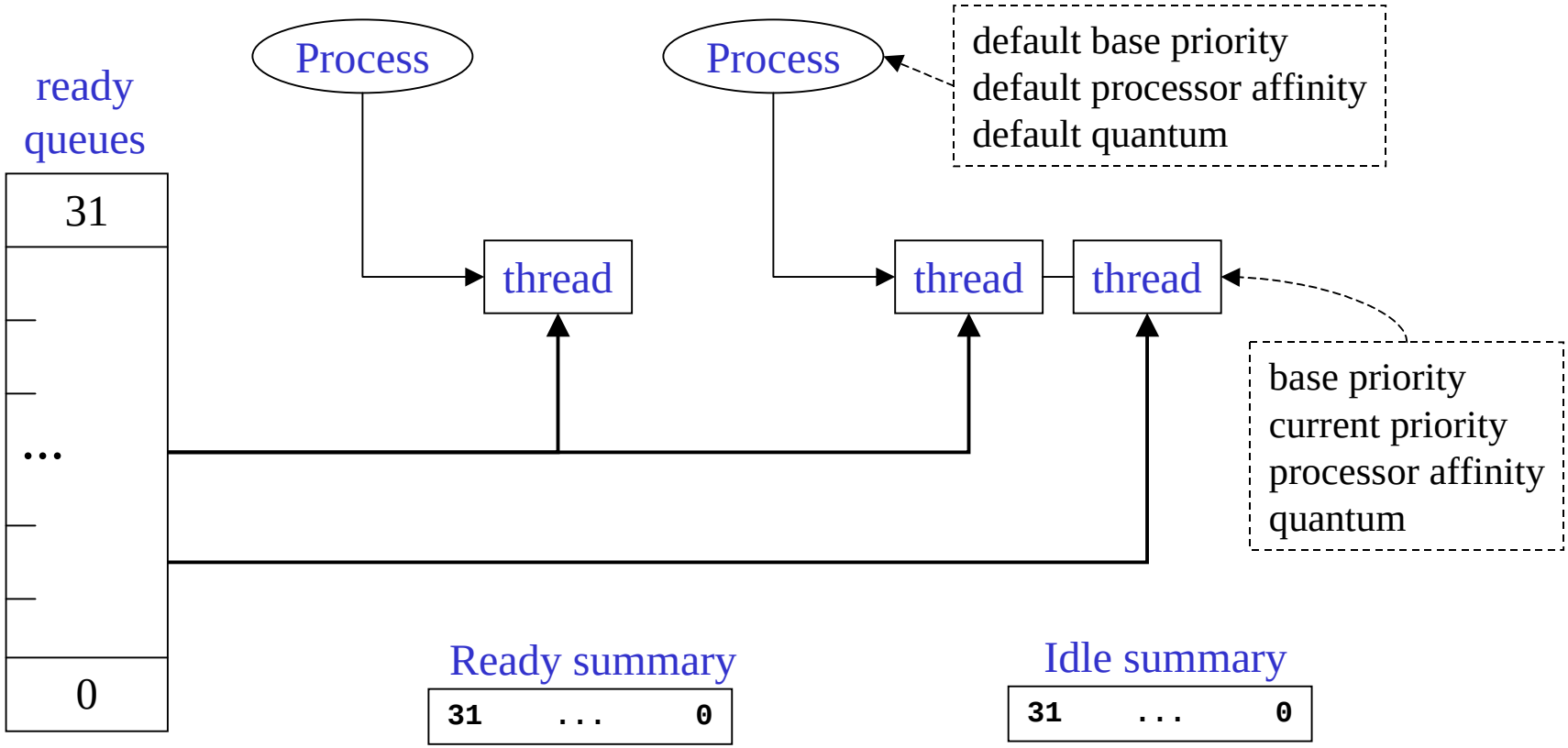
CSE 410 - Computer Systems

November 21, 2001

Readings and References

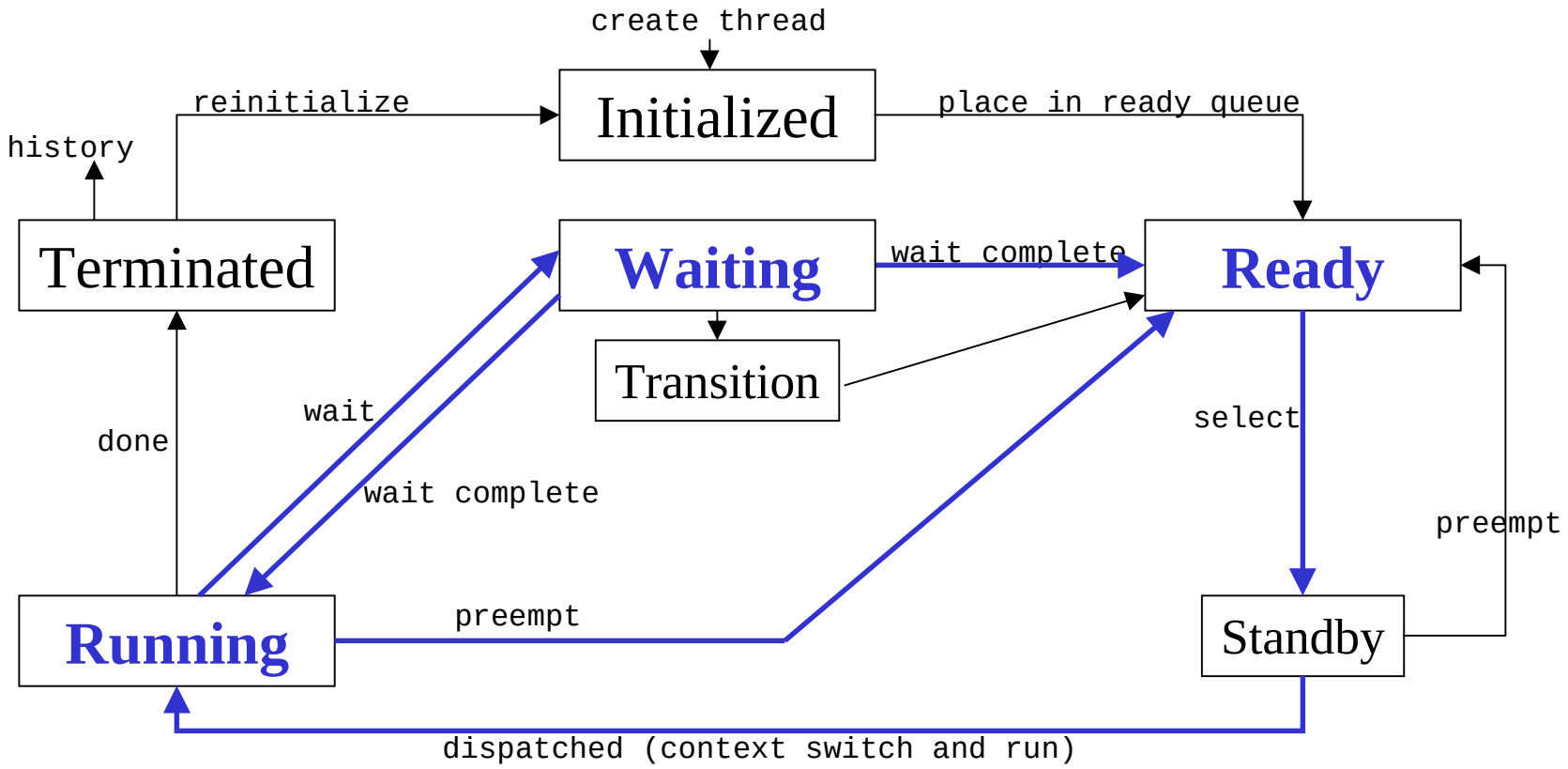
- Reading
  - Chapter 6, Section 6.7.2, *Operating System Concepts*, Silberschatz, Galvin, and Gagne
- Other References
  - Chapter 6, Section “Thread Scheduling”, *Inside Microsoft Windows 2000*, Third Edition, Solomon and Russinovich. This book is the source of most of today’s lecture.
  - Chapter 6, Performance Monitoring, *Windows 2000 Professional Resource Kit*, Microsoft

Dispatcher “database”



clockres.exe

Thread State Transitions

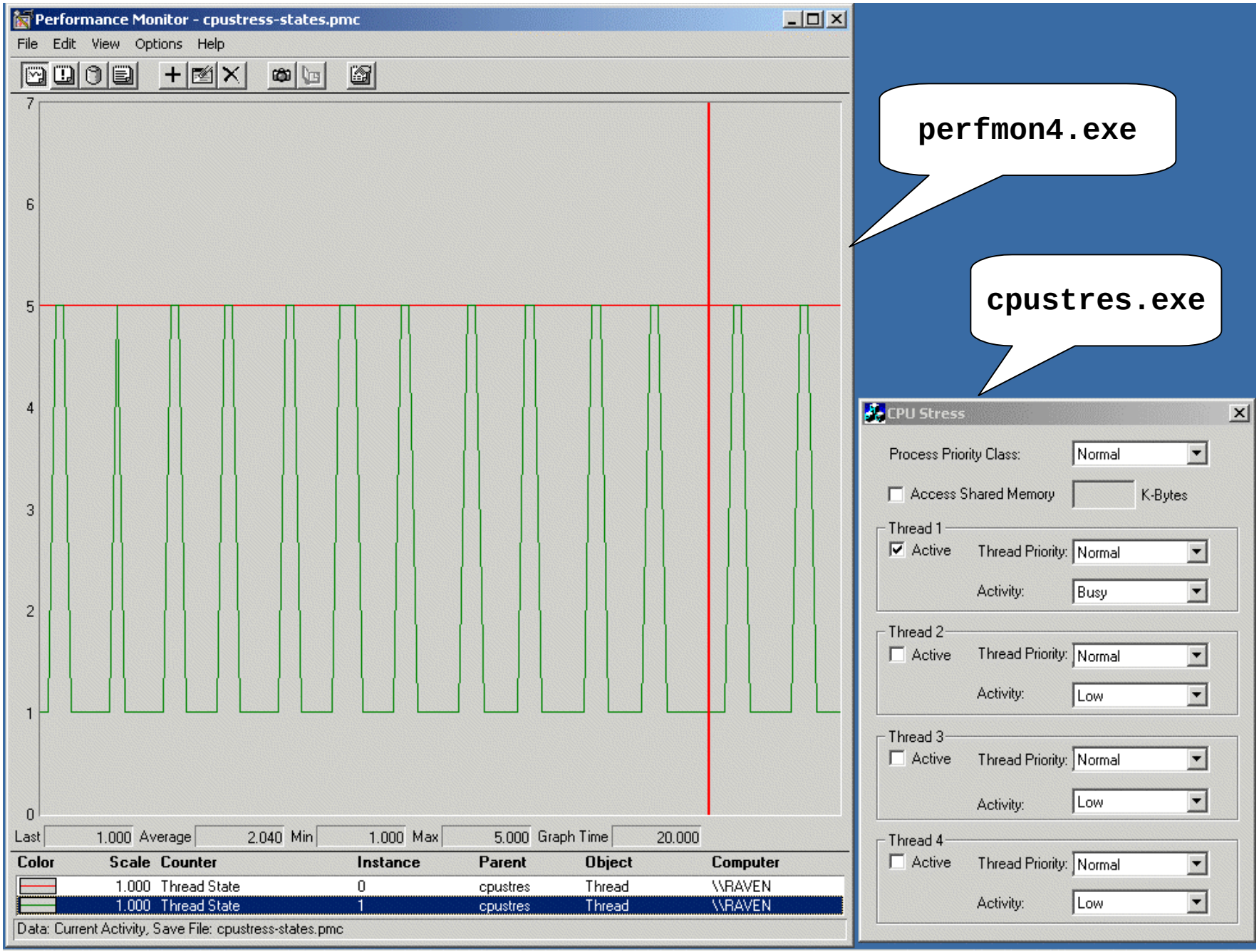


Ready, Running, Waiting

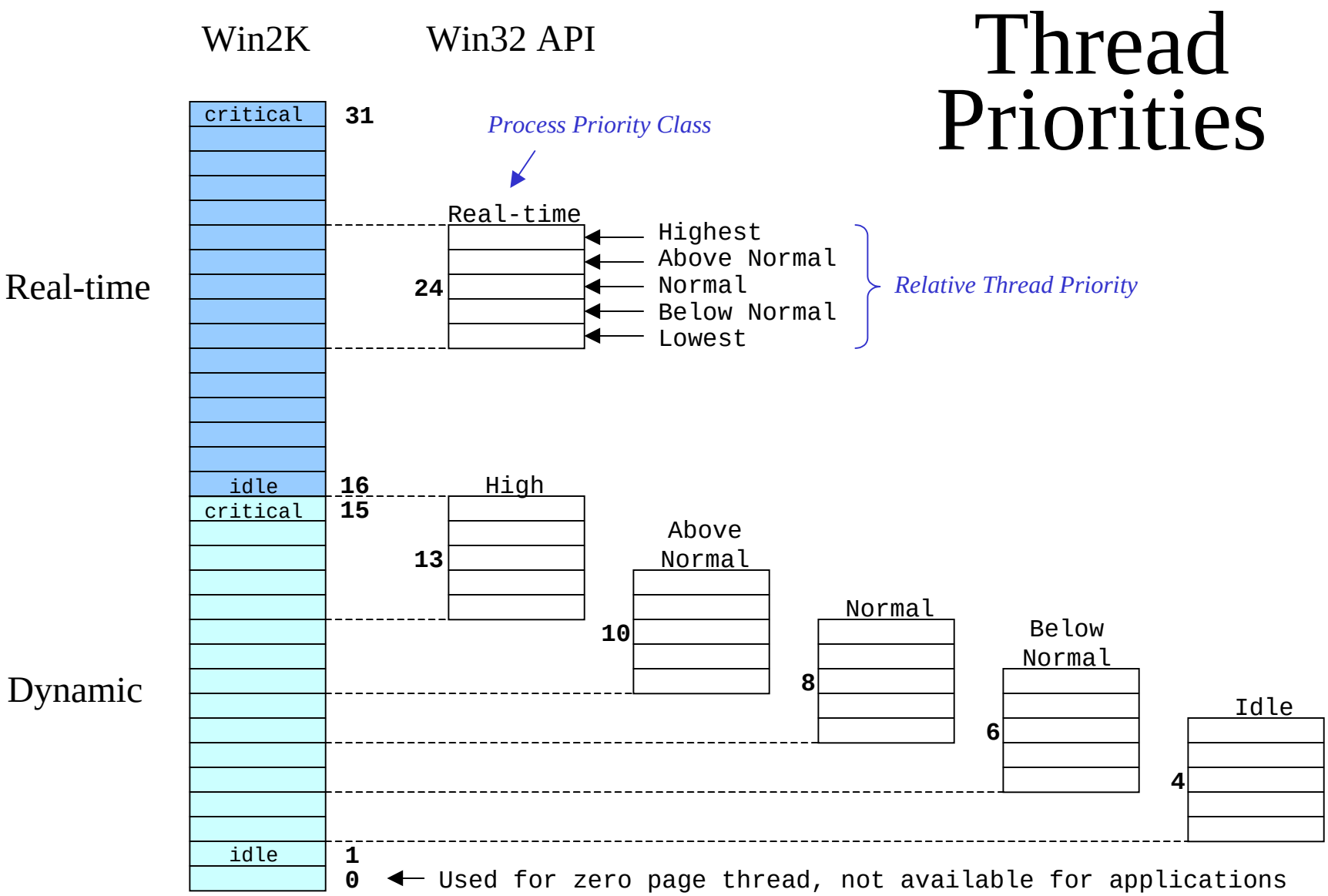
- Ready
  - › ready to run if there is a processor available
  - › there is a ready queue for each priority level
- Running
  - › has been switched to and is running
- Waiting
  - › waiting on an event (synchronize, I/O, etc)

Other States

- Initialized
  - On its way in the door
- Terminated
  - On its way out the door to history or recycle
- Standby
  - Ready and selected to run next
- Transition
  - Ready, but important parts are paged out


Performance Monitor - cpustress-states.pmc
File Edit View Options Help
7
6
5
4
3
2
1
0
Last 1.000 Average 2.040 Min 1.000 Max 5.000 Graph Time 20.000
Color Scale Counter Instance Parent Object Computer
1.000 Thread State 0 cpustres Thread \\RAVEN
1.000 Thread State 1 cpustres Thread \\RAVEN
Data: Current Activity, Save File: cpustress-states.pmc
perfmon4.exe
cpustres.exe
CPU Stress
Process Priority Class: Normal
Access Shared Memory K-Bytes
Thread 1
Active
Thread Priority: Normal
Activity: Busy
Thread 2
Active
Thread Priority: Normal
Activity: Low
Thread 3
Active
Thread Priority: Normal
Activity: Low
Thread 4
Active
Thread Priority: Normal
Activity: Low

Thread Priorities

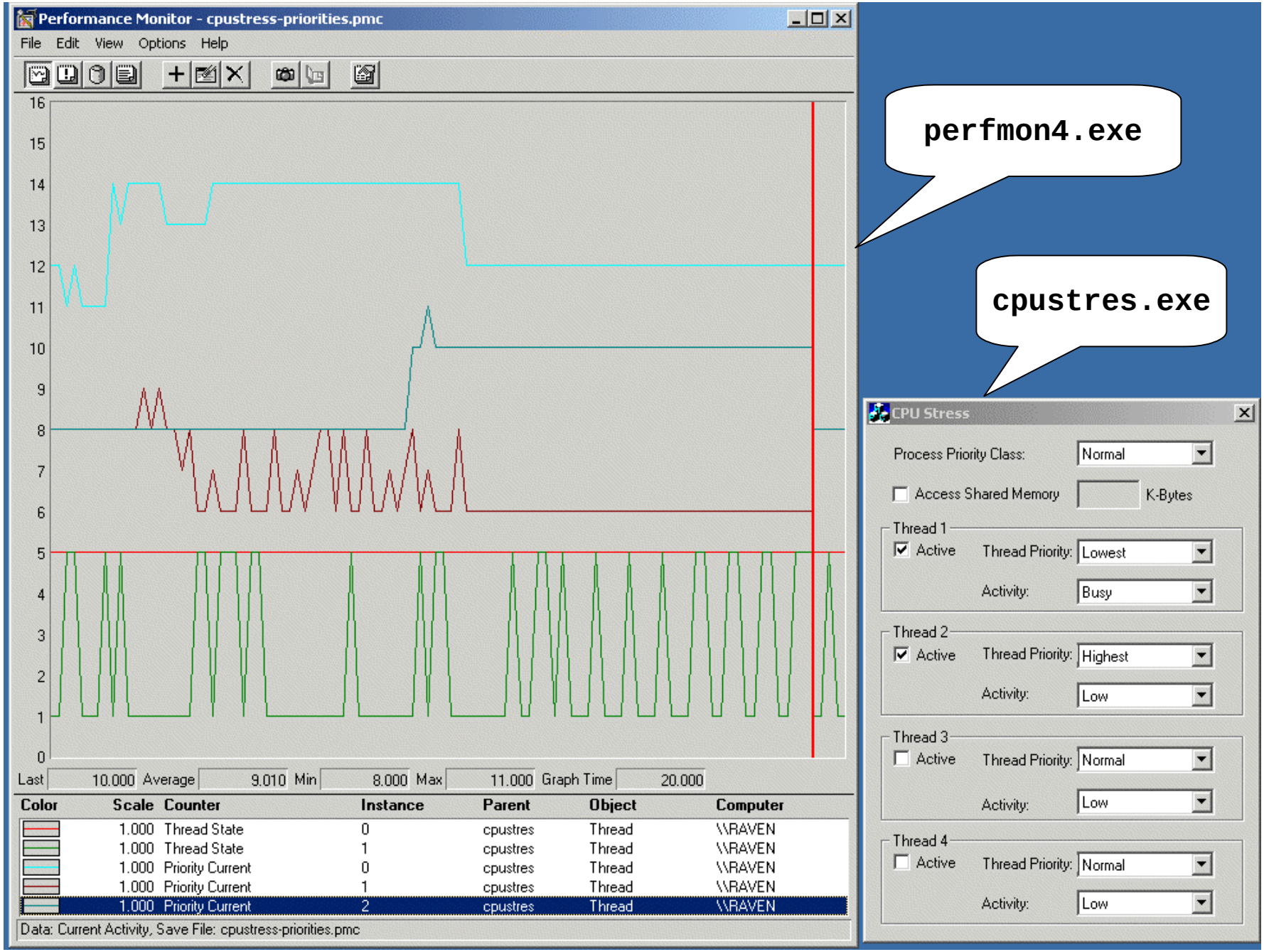


Setting Thread Priorities

- Base priority
  - normally inherited from process default
  - can be explicitly set
- Current priority
  - starts out same as base
  - real time never changes
  - dynamic is boosted when appropriate for responsiveness

Priority boosting

- After I/O completion or event wait
  - › you've waited for this data, now use it quick
- User response
  - › Foreground thread after a wait or window thread wakeup for window event
- CPU starvation
  - › found an aging thread on the ready queues
- The boost decays quickly over time


Performance Monitor - cpustress-priorities.pmc
File Edit View Options Help
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Last 10.000 Average 9.010 Min 8.000 Max 11.000 Graph Time 20.000
Color Scale Counter Instance Parent Object Computer
1.000 Thread State 0 cpustres Thread \\RAVEN
1.000 Thread State 1 cpustres Thread \\RAVEN
1.000 Priority Current 0 cpustres Thread \\RAVEN
1.000 Priority Current 1 cpustres Thread \\RAVEN
1.000 Priority Current 2 cpustres Thread \\RAVEN
Data: Current Activity, Save File: cpustress-priorities.pmc
perfmon4.exe
cpustres.exe
CPU Stress
Process Priority Class: Normal
Access Shared Memory K-Bytes
Thread 1
Active
Thread Priority: Lowest
Activity: Busy
Thread 2
Active
Thread Priority: Highest
Activity: Low
Thread 3
Active
Thread Priority: Normal
Activity: Low
Thread 4
Active
Thread Priority: Normal
Activity: Low

Quantum

- Thread Quantum is
  - › indicator of the amount of time a thread can run before W2K checks whether another thread at the same priority should get to run
- Each thread has a current quantum value
  - › a small integer that is decremented under various circumstances
  - › not an actual length of time, just a number

Quantum value

- Thread quantum is initialized when thread is put on the ready queue
  - initial value of 6 on Windows 2K Professional
  - initial value of 36 on Windows 2K Server
- Quantum of running thread is decremented by 3 after system clock interrupt
  - so a W2K Pro thread can run for 2 clock intervals
  - a W2K Server thread can run for 12 clock intervals

Quantum is reset to initial value

- a thread moves to ready queue after quantum end
  - in other words, a thread is given another chunk of time to use after it has exhausted the first chunk
- a real-time thread is preempted and moves from running to ready or it moves from running to wait
  - the presumption is that you are doing a good job of explicitly managing priorities and access to the CPU when you are running real-time threads

Quantum changes

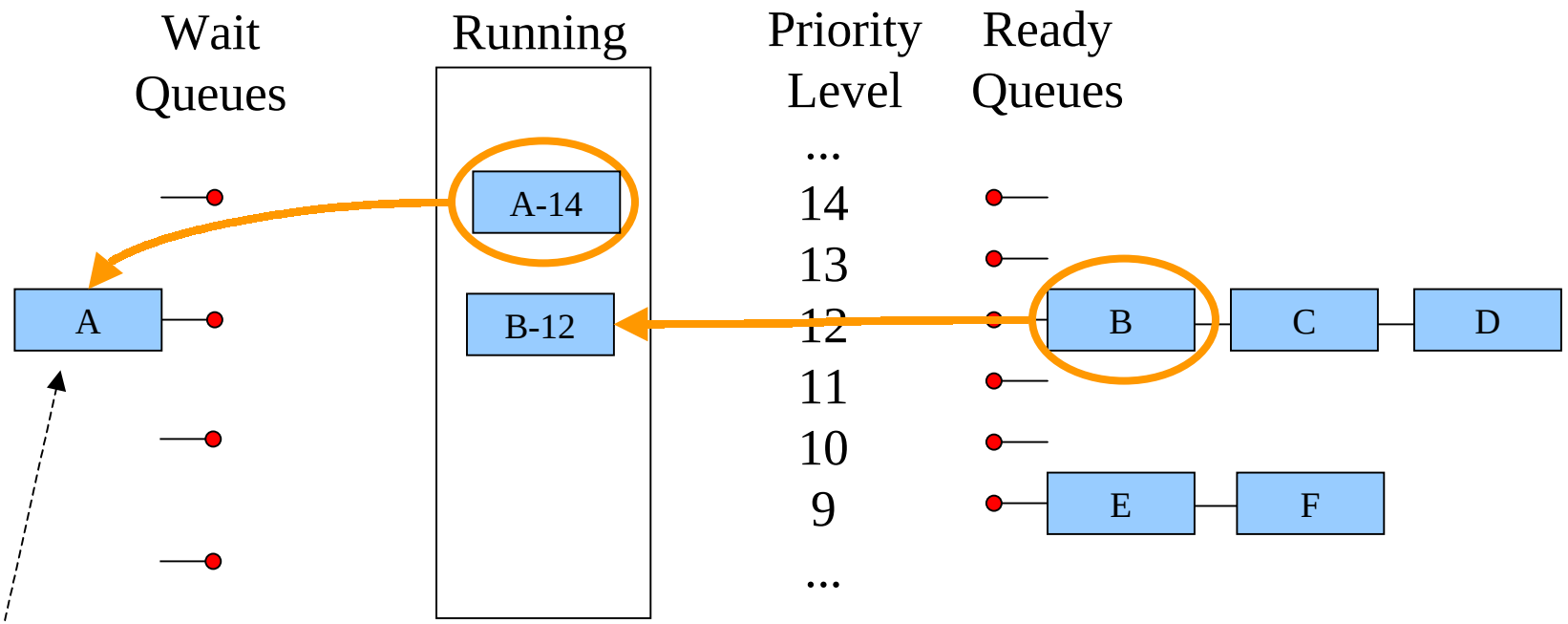
- Quantum is decremented
  - › reduced quantum => less time remaining before thread has exhausted its time slice
  - › reduced by 3 when the clock ticks
  - › by 1 when dynamic thread executes a wait
- Quantum initial value may be boosted
  - › “Optimize performance for applications”

  => boost initial quantum for foreground threads

Scheduling Scenarios

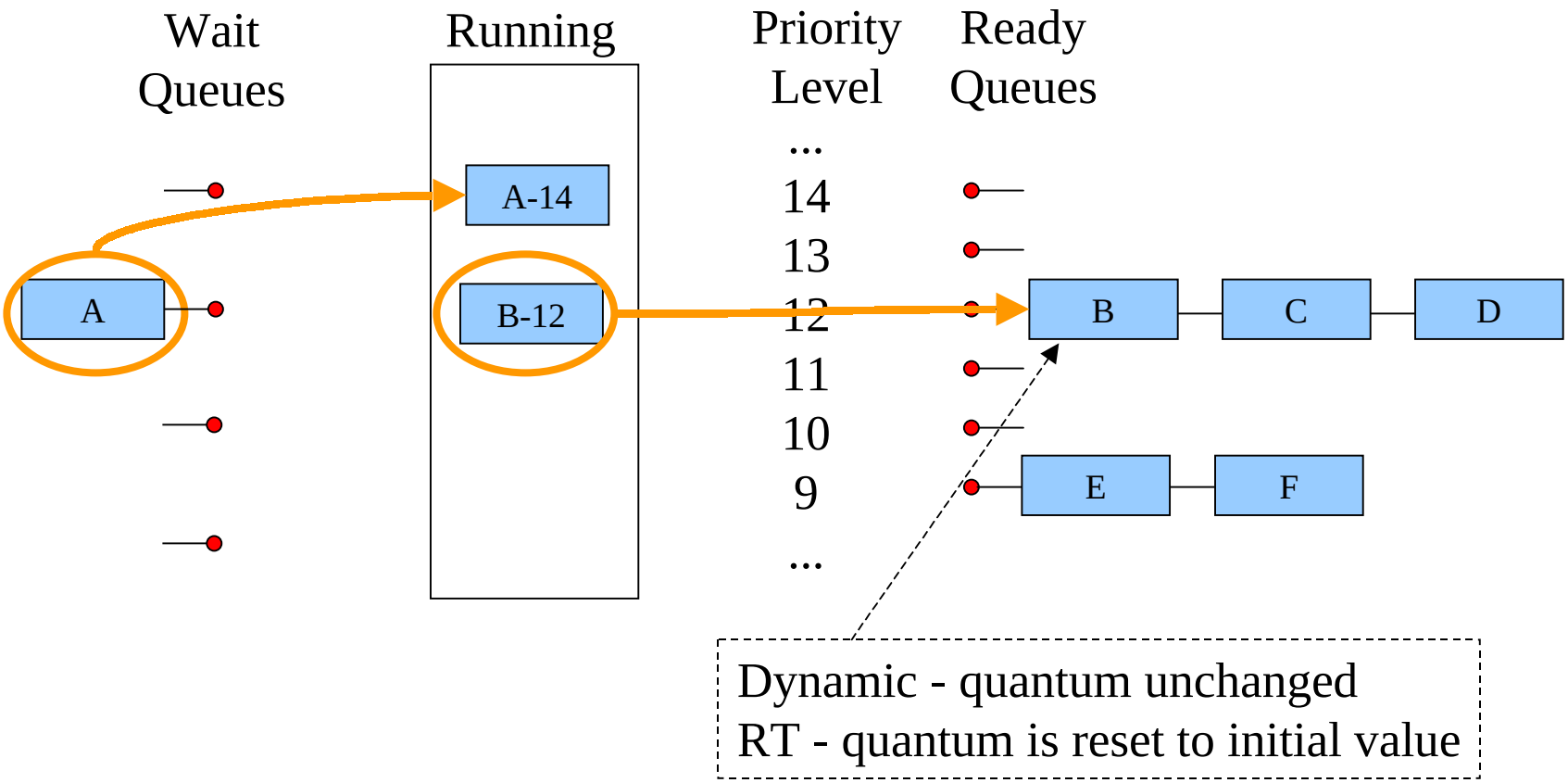
- Voluntary switch
  - thread calls a wait function of some sort
- Preemption
  - higher priority thread is ready to run
- Quantum end
  - the running thread exhausts its quantum

Voluntary Switch

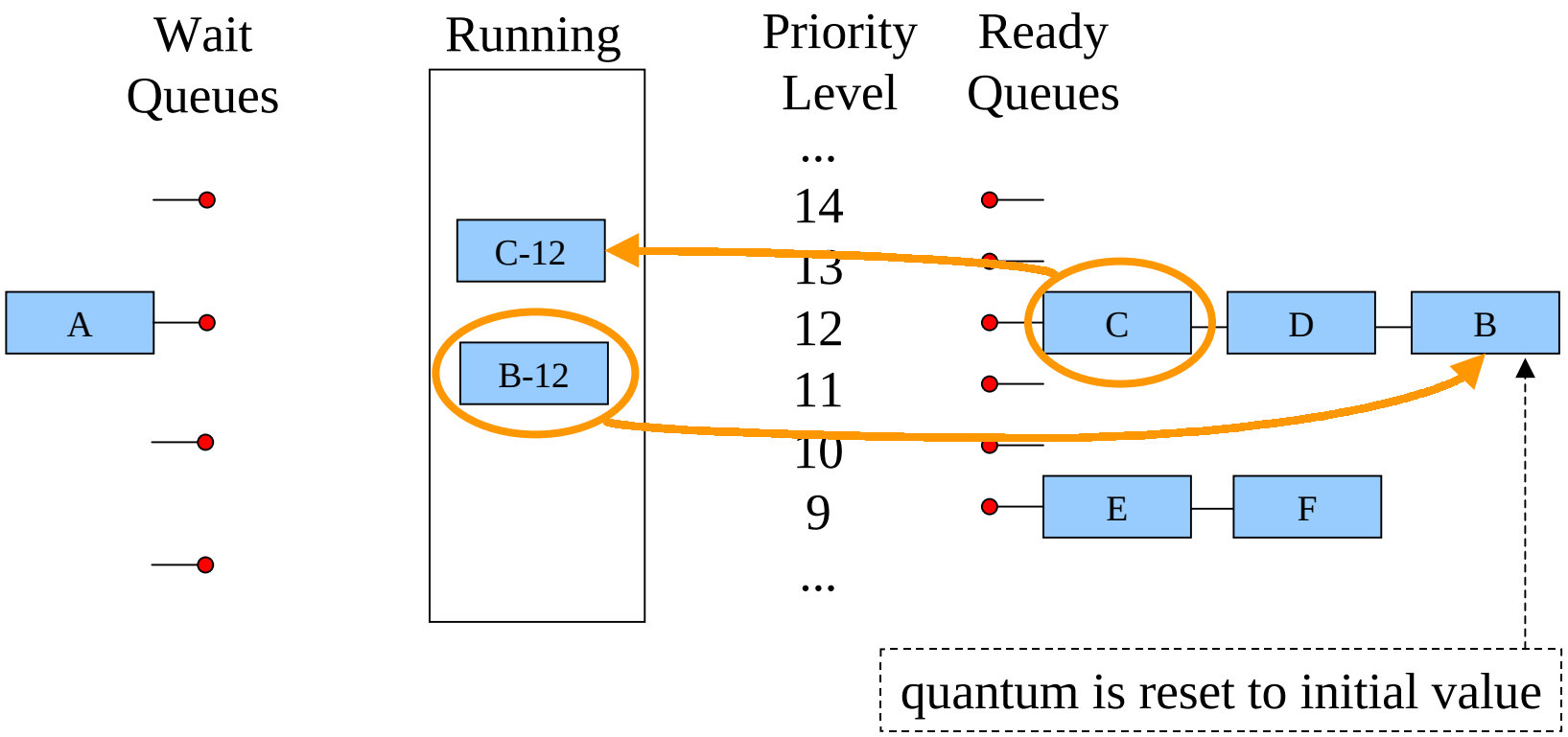


Dynamic - quantum is decremented by 1
RT - quantum is reset to initial value

Preemption


Wait Queues
Running
Priority Level
Ready Queues
...
14
13
12
11
10
9
...
A-14
B-12
A
B
C
D
E
F
Dynamic - quantum unchanged
RT - quantum is reset to initial value

Quantum End


Wait Queues
Running
Priority Level
Ready Queues
...
14
13
12
11
10
9
...
A
C-12
B-12
C
D
B
E
F
quantum is reset to initial value