SQL Server: Performance Troubleshooting Using Wait Statistics

Module 2: SQL Server Threading Model

Paul S. Randal
Paul@SQLskills.com

pluralsight
hardcore developer training

Introduction

- **SQL Server is really a mini operating system**
- **It performs and controls its own:**
  - Memory management
  - I/O
  - Scheduling

- **In this module we'll cover:**
  - Threads
  - Scheduling
  - Scheduling states and state transitions
  - Special cases

What Are Threads?

- **A thread is unit of execution within a process**
- **Multiple threads can exist within a process**
  - Each is given small amount of processor time, then waits while other threads execute (called scheduling)
  - This gives the impression of operations happening concurrently
  - See this Wikipedia link for more background: http://bit.ly/Mx94C4

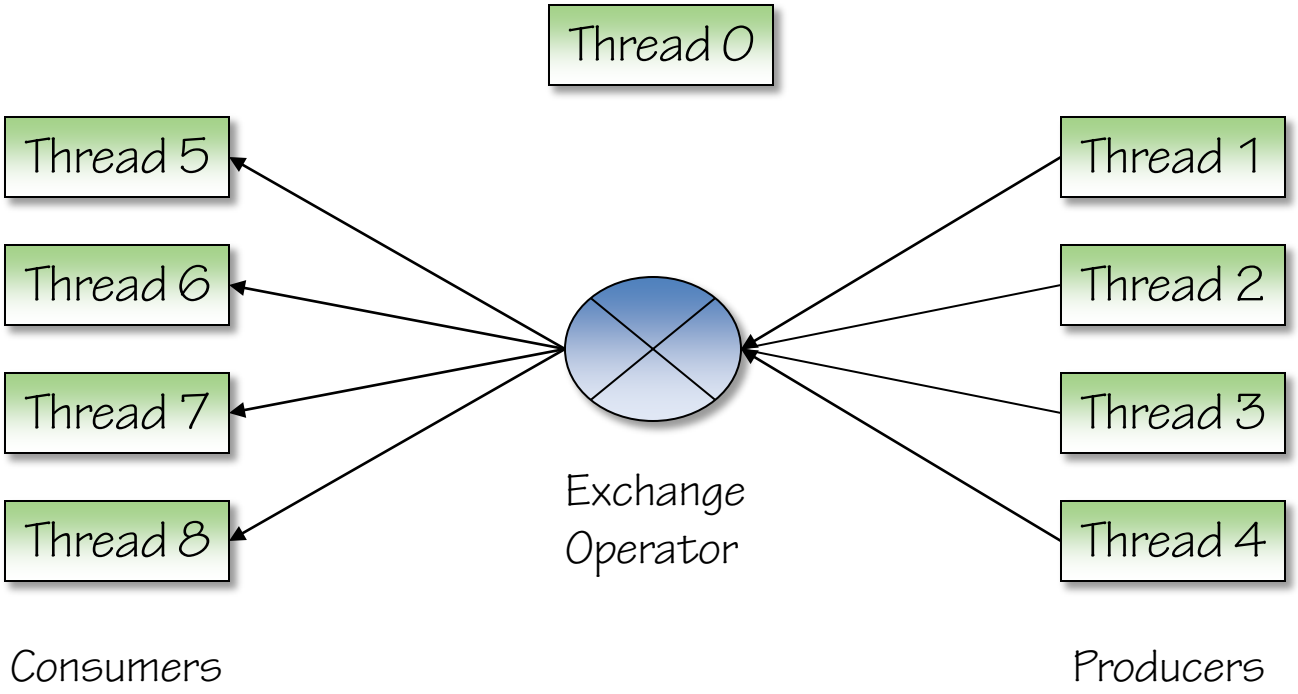
- **SQL Server uses operating system threads to perform its work**
  - Called worker threads
  - Some are dedicated threads, such as for performing checkpoints, deadlock monitoring, and ghost record cleanup
  - Most are in a pool of threads that SQL Server uses to process user requests

Parallelism

- **The Query Processor may determine that a particular operation can be performed more efficiently by using multiple threads**
  - Called running in parallel
  - Multiple threads execute a small portion of the operation at the same time
- **In that case, the Query Optimizer will produce a query plan that can be parallelized**
  - Otherwise it generates plan that can only run serially (i.e. single-threaded)
- **At execution time, the Query Execution portion of the Query Processor decide what parallelism to use**
  - If the plan is serial only, no parallelism is used
  - If the plan can be run in parallel, the decision is made on how far to parallelize based on available resources, and a variety of configuration options

Parallel Threads Example

- **As part of a query plan, you may see the**  **operator, for example**
- **This is a Repartition Streams operation**
  - Uses producer and consumer threads, plus a control thread
- **For a degree-of-parallelism = 4 operation, the threads would look like:**



Controlling Parallelism

- **By default, the server-wide limit on parallelism is disabled**
  - The sp_configure option 'max degree of parallelism' is set to '0'
  - SQL Server decides how far to parallelize each query
- **If configured to anything except '0', this limit can be over-ridden**
  - Use the OPTION MAXDOP (x) syntax, where 'x' is the degree of parallelism requested (from 0 to 64)
  - Any user with any privilege can do this
- **In Enterprise Edition of SQL Server 2008 onwards, Resource Governor can be used to provide a hard limit to the degree of parallelism a query can utilize**
  - Using the MAX_DOP option for a workload group
  - This cannot be over-ridden in any way

- **Specifying a degree of parallelism does not guarantee it will be used**

Thread Scheduling

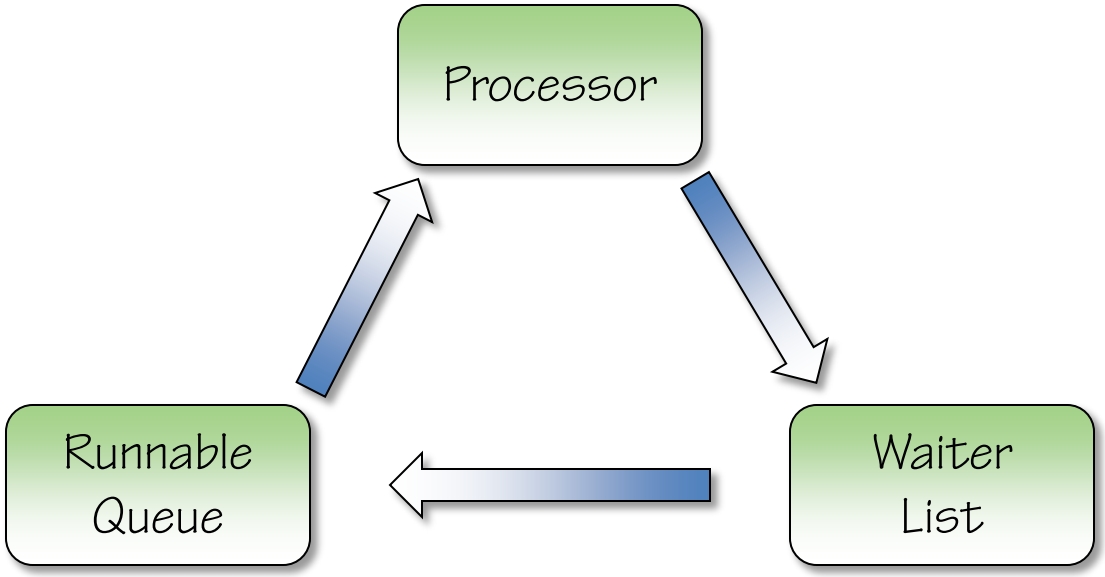
- **SQL Server performs its own thread scheduling**
  - Called non-preemptive scheduling
  - More efficient (for SQL Server) than relying on Windows scheduling
  - Performed by the SQLOS layer of the Storage Engine

- **Each processor core (whether logical or physical) has a scheduler**
  - A scheduler is responsible for managing the execution of work by threads
  - Schedulers exist for user threads and for internal operations
  - Use the sys.dm_os_schedulers DMV to view schedulers

- **When SQL Server has to call out to the OS, it must switch the calling thread to preemptive mode so the OS can interrupt it if necessary**
  - More information on this in Module 5

Components of a Scheduler

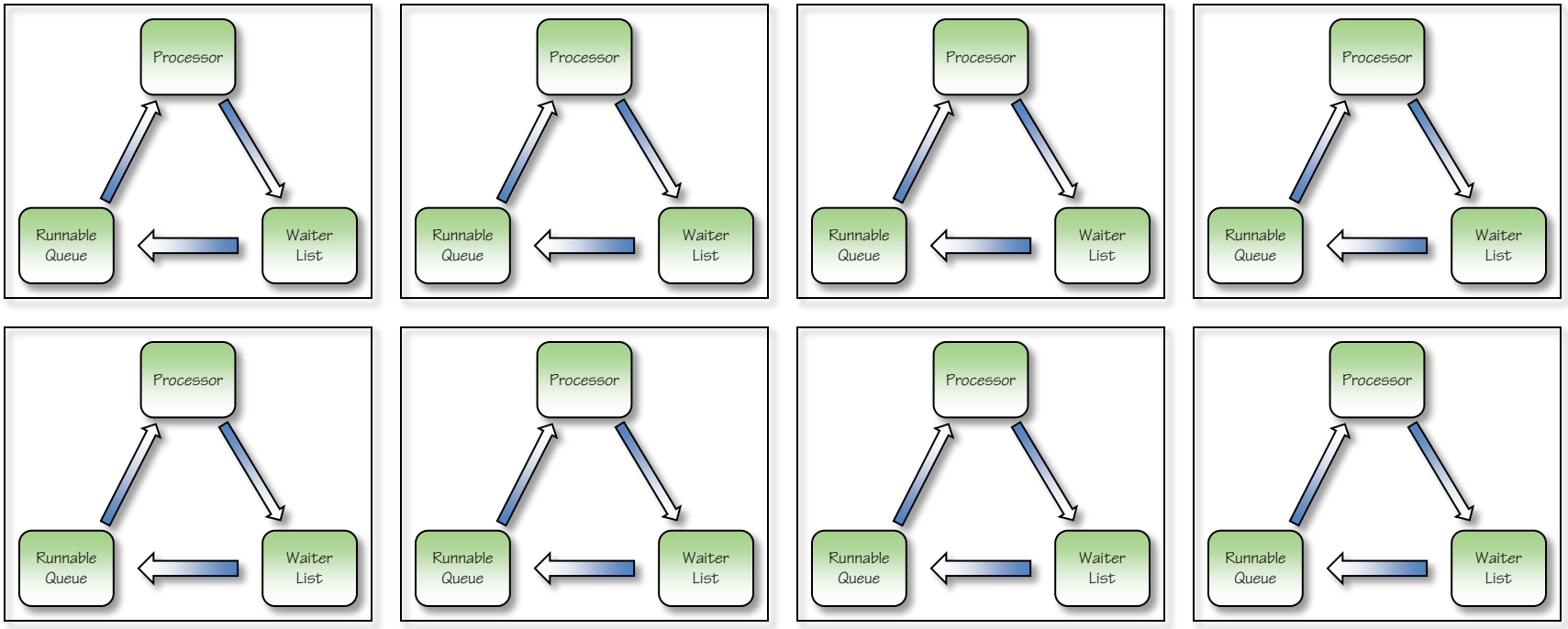
- All schedulers are composed of three 'parts'



- Threads transition around these parts until their work is complete

Schedulers in SQL Server

- **One scheduler per logical or physical processor core**
  - Plus some extra ones for internal tasks and the Dedicated Admin Connection
- **For example, for a server with four physical processor cores, with hyper-threading enabled, there will be eight user schedulers**



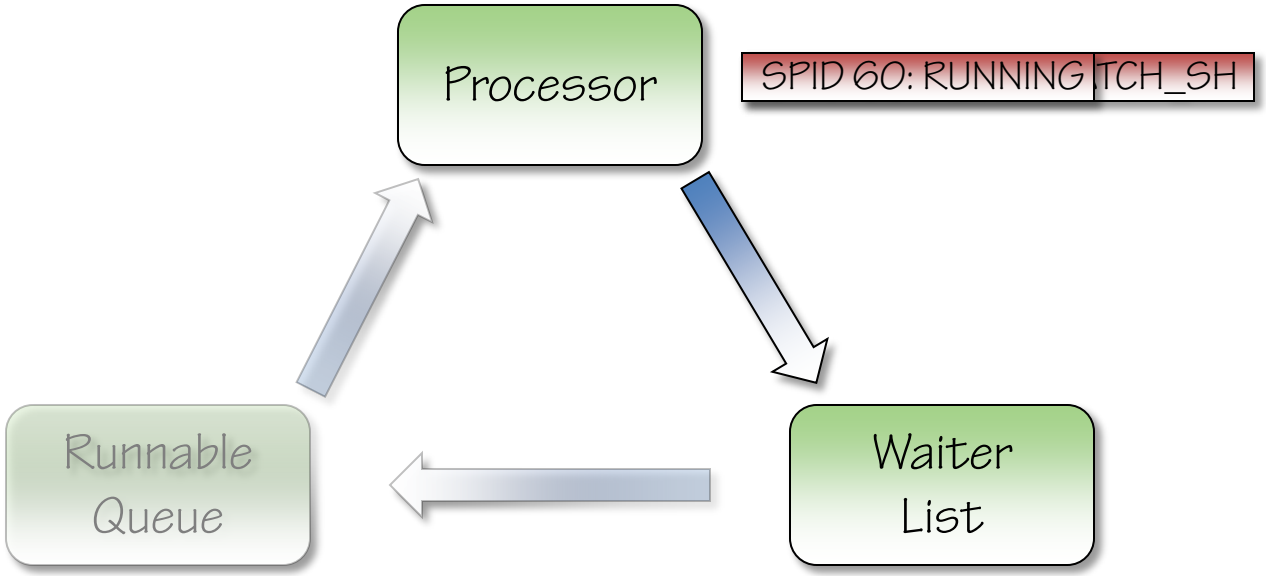
Thread States

- **A thread can be in one of three states when being actively used as part of processing a query**
- **RUNNING**
  - The thread is currently executing on the processor
- **SUSPENDED**
  - The thread is currently on the Waiter List waiting for a resource
- **RUNNABLE**
  - The thread is currently on the Runnable Queue waiting to execute on the processor

- **Threads transition between these states until their work is complete**

Transition from RUNNING to SUSPENDED

- **A thread continues executing on the processor until it must wait for a resource to become available**
  - The thread's state changes from RUNNING to SUSPENDED
  - The thread moves to the Waiter List
  - This process is called being 'suspended'

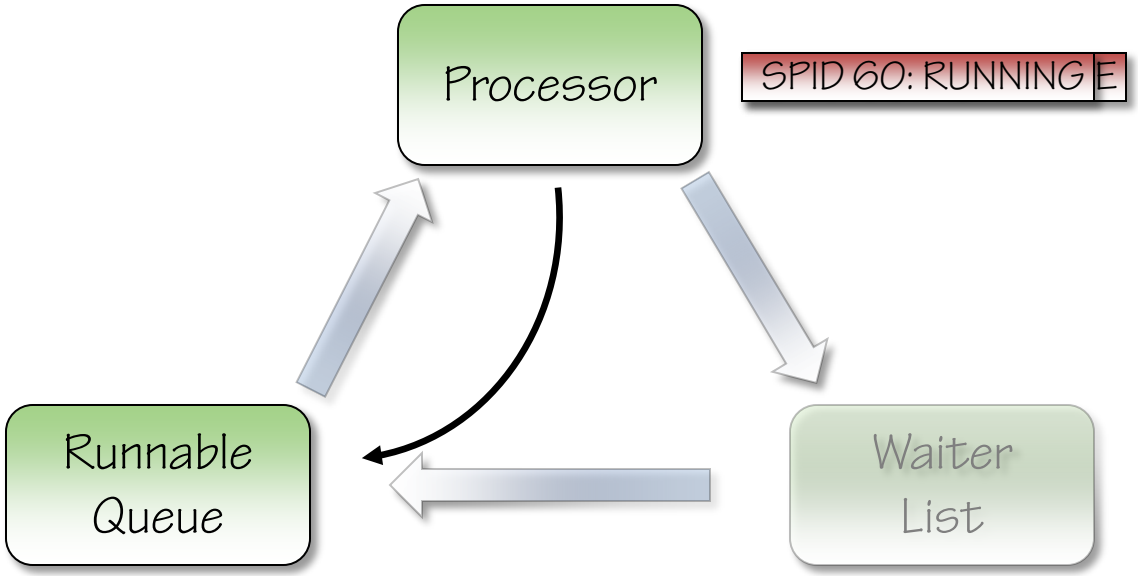


The Waiter List

- **The Waiter List is an unordered list of threads that are suspended**
- **Any thread can be notified at any time that the resource it is waiting for is now available**
  - Again, absolutely no ordering
- **There is no limit to how long a thread remains on the waiter list**
  - Although execution timeouts or lock timeouts may take effect
- **There is no practical limit to how many threads can be on a scheduler's waiter list at any time**
- **The sys.dm_os_waiting_tasks DMV shows which threads are currently waiting and what they are waiting for**
  - More details on this in Module 3

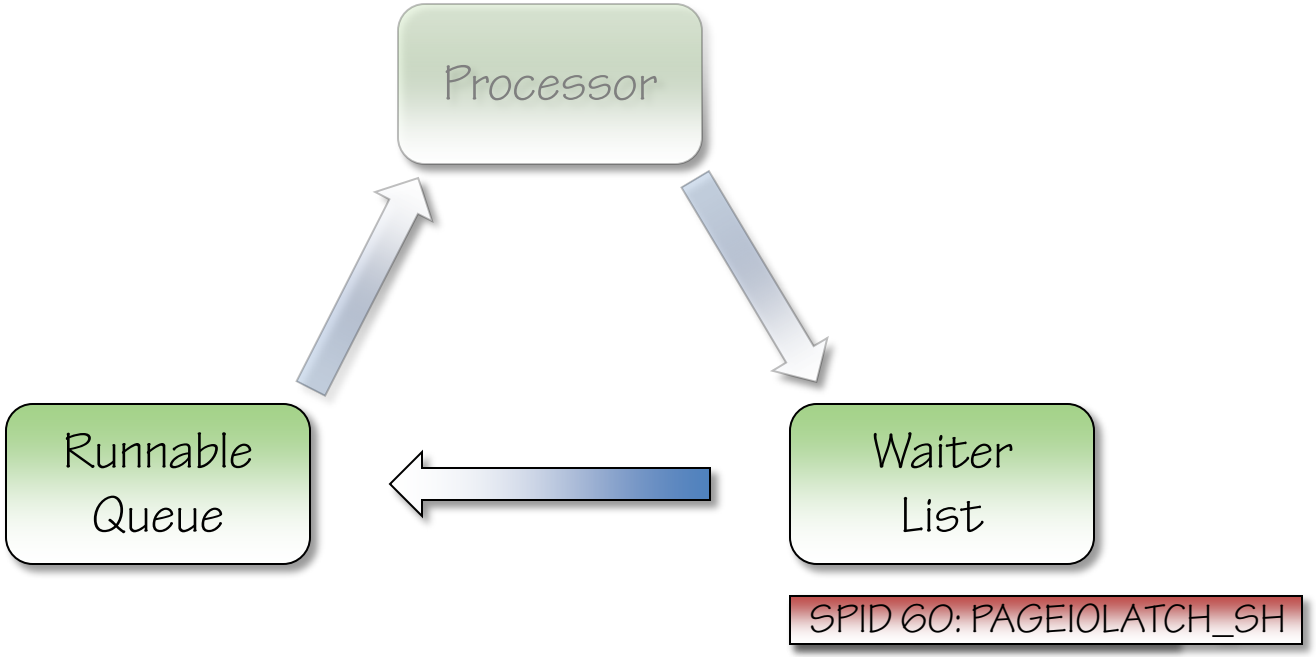
Special Case: Quantum Exhaustion

- **If a thread does not need to wait for a resource, it will continue executing until its quantum is exhausted**
  - Thread quantum is fixed at 4 milliseconds and cannot be changed
- **If this occurs the thread moves to the bottom of the Runnable Queue**
  - The thread's state changes from RUNNING to RUNNABLE
  - More information on this in Module 5



Transition from SUSPENDED to RUNNABLE

- **A thread continues to wait until it is told that the resource is available**
  - The thread's state changes from SUSPENDED to RUNNABLE
  - The thread moves to the bottom of the Runnable Queue
  - This process is called being 'signaled'



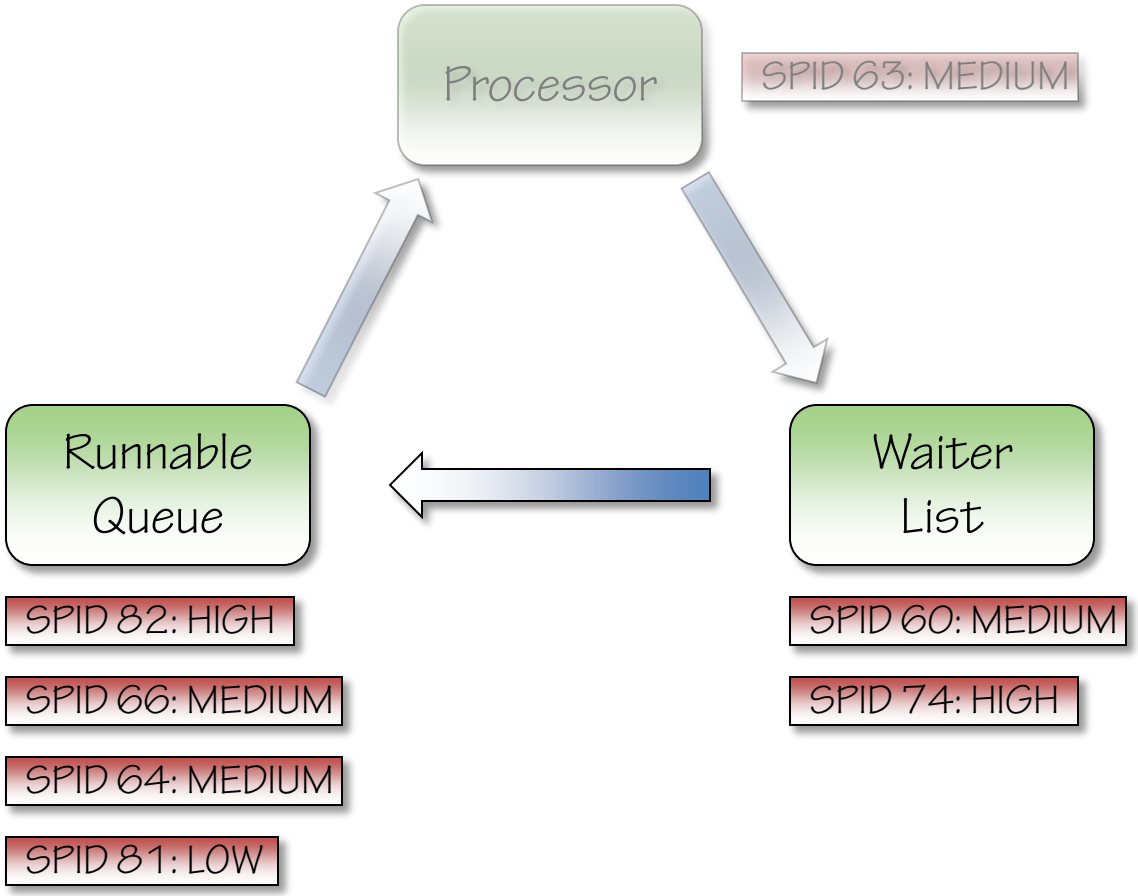
The Runnable Queue

- **The Runnable Queue is a strict First-In-First-Out (FIFO) queue**
  - There is a special case that will be discussed on the next slide
- **Threads enter the queue at the bottom and progress to the top**
- **The thread at the top of the queue is the one that will execute on the processor when the processor becomes free**
  - When the currently executing thread is suspended or exhausts its quantum
- **The size of the Runnable Queue can be seen from the runnable_tasks_count column in sys.dm_os_schedulers**

Special Case: Resource Governor

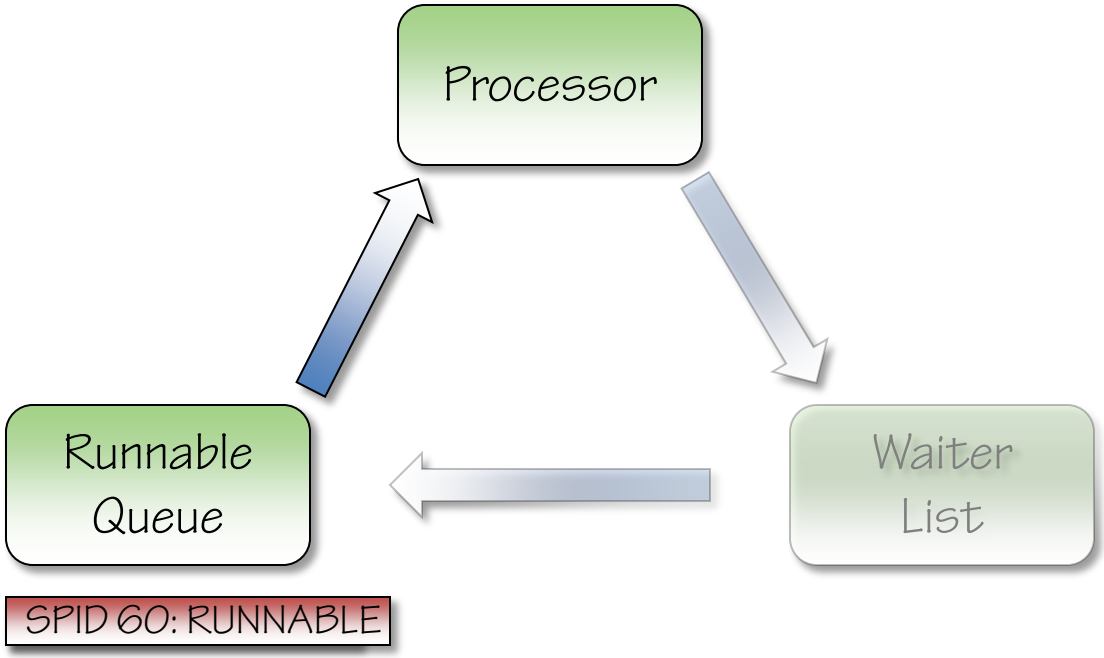
- **Using Resource Governor (in Enterprise Edition of SQL Server 2008 onwards) it is possible to define multiple workload groups that share the same resource pool**
  - A workload group can be thought of as a bucket of connections
  - A resource pool can be thought of as a set of CPU and memory limits, but more details are beyond the scope of this course
- **Multiple workload groups for a resource pool can be assigned relative priorities**
  - The default priority is medium
  - Possible values are high, medium, and low
- **The relative priorities change how the Runnable Queue works**
  - High to medium to low equates to 9 to 3 to 1 in terms of the priority of the threads that are permitted to execute
  - For example, for each low-priority thread on the runnable queue, nine high-priority threads will be allowed to jump over it and execute first

Resource Governor Example

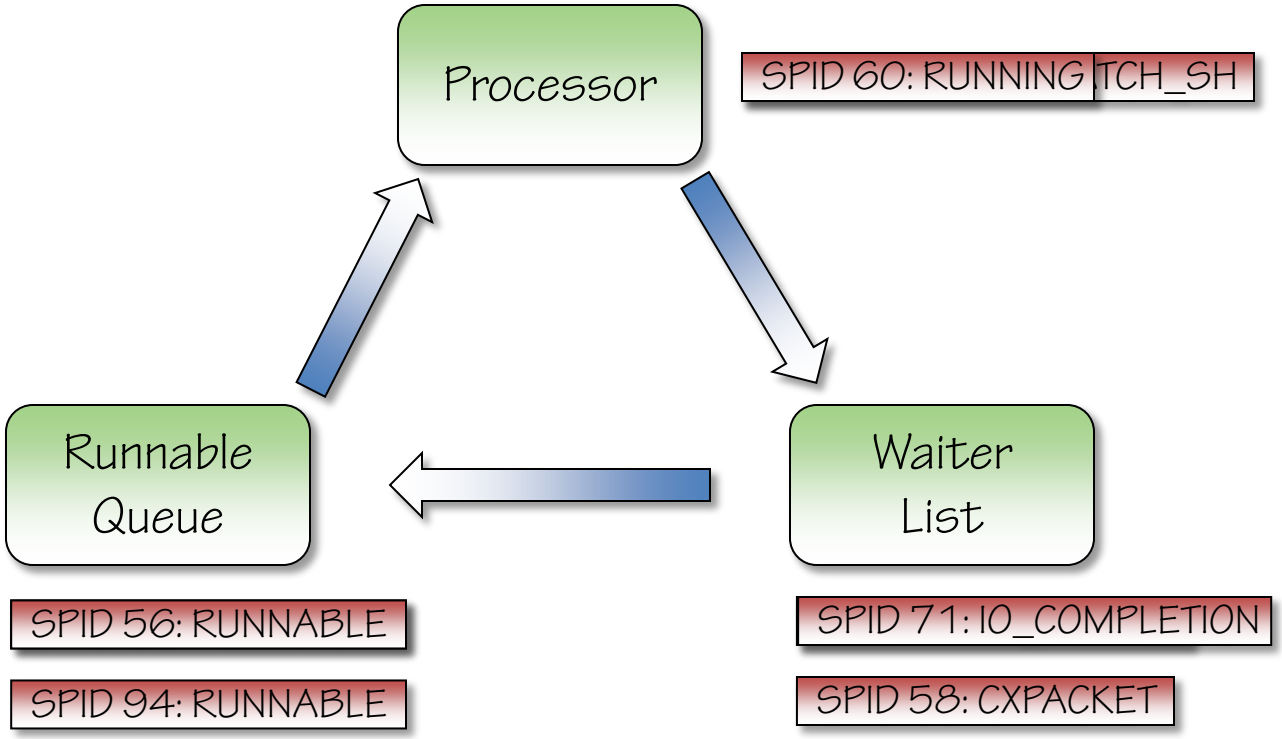


Transition from RUNNABLE to RUNNING

- **The thread waits on the Runnable Queue until it reaches the top and the processor becomes available**
  - The thread's state changes from RUNNABLE to RUNNING



Pulling It All Together



Summary

- **SQL Server performs its own thread scheduling**
- **Threads move from RUNNING to SUSPENDED to RUNNABLE and back to RUNNING again until their work is completed**
  - This can be altered somewhat using Resource Governor
- **Analysis of the reasons for a thread going from RUNNING to SUSPENDED and how long a thread remains SUSPENDED is the basis of the waits and queues methodology**
- **The next module will define the various wait times and how to extract them from SQL Server**