Microsoft Office Project Server 2007 –
Performance and Capacity Planning Best Practices

Prepared for

EPM Community Quality Assurance

July, 2009

Version 1.1

Prepared by

World Wide Center of Excellence for EPM (WW EPM COE)

Microsoft | Services

Table of Contents

1. Introduction 3

1. Purpose 3

2. Audience 3

3. Scope 3

4. Additional Factors 4

2. How to Use This Document 6

3. Container and Object Best Practices 7

1. Container: Platform 7

   A. Object: Farm 7

   B. Object: Shared Services Provider 10

   C. Object: Application Pool 11

   D. Object: Web Application 12

   E. Object: Database 13

   F. Object: System - Disk 16

   G. Object: System - Network 18

   H. Object: System - Memory 19

   I. Object: System - CPU 19

   J. Object: Logs 20

   K. Object: Performance Counters 21

2. Container: Data 26

   A. Object: Projects 26

   B. Object: Tasks 29

   C. Object: Assignments 31

   D. Object: Resources 33

E. Object: Custom Fields 35
F. Object: Security 37
3. Container: User 39
A. Object: Localization 39
B. Object: Workloads 40
C. Object: Queue Job Processor Threads 42
D. Object: Interface Feeds 44
E. Object: System 45
4. Sizing Calculation Factors 47
1. Sample Data Scalar Values 47
2. Application Server sizing 48
3. SQL Server sizing 49
4. Web Front End sizing 50
Appendix 52
Available Project Server 2007 solutions 52
A. Performance Counters 55
B. Unified Logging Service (ULS) - Logs 60
C. Internet Information Services (IIS) - Logs 61
D. Windows Server Event Log 62

1. Introduction

1. Purpose

The purpose of this guide is to extend the Microsoft Office Project Server 2007 Performance Testing Lab (http://www.microsoft.com/downloads/details.aspx?FamilyID=2f595e22-c9f1-4376-a8b2-7a7eccd2aba5&displaylang=en) whitepaper by providing Best Practices and Recommendations. Further recommendations are made in this guide based on field experience and deployment successes in real world scenarios against the main areas of the system. The emphasis is on Project Server 2007 in a production environment using physical servers. Reference is made to virtualization in the relevant sections. However, for a more comprehensive overview please read the Project Server 2007 Hyper-V guide (http://technet.microsoft.com/en-us/library/cc982157.aspx).

The guidelines and recommendations in this guide present the "what" and "why". Certain areas are prescriptive providing steps on "how" to accomplish the recommendation. However, the assumption is the readers of this guide are able to understand the reasons behind the guidance and take the necessary actions within their organization to implement optimization that applies to their scenario.

The guidelines and recommendations in this guide cover technical, functional and process related guidance. It's important to note these are mutually inclusive of each other and present cross-impacts through the system when a design is not balanced across all 3 areas. It's expected that administrators of a Project Server 2007 deployment will collect regular operational and functional data, and over time trend the system for performance and scalar depth. Optimization in all areas is required on a regular basis to maintain an efficient system and identify remediation steps through operational policies.

For more information about performance testing, see Performance and capacity planning best practices on TechNet (http://go.microsoft.com/fwlink/?LinkId=150595).

2. Audience

The intended audience includes IT administrators, database administrators, data architects, system engineers, project managers and business analysts who may be involved in implementing Project Server 2007. This document is designed for a reader who has general knowledge of all the required application and platform components that support Project Server 2007.

3. Scope

What it's not: This document is not a "How-to" guide.

What it is: The content provides insight to "What" and "Why" applied best practices for users to plan the most effective implementation within their organization. Refer to TechNet http://technet.microsoft.com/en-us/office/projectserver/default.aspx for content on the "How-to" and to the Appendix with links to additional solutions.

The guidelines and recommendations represent how organizations use Project Server 2007 successfully. The recommendations are not limits or restrictions. They are thresholds to be observed with the expectation, that exceeding the threshold, may result in gradual performance degradation in a certain area, either accepting this tradeoff or taking remediation steps to bring the system back within the desired design. Project Server 2007 is a data intensive application coupled with calculations and transformation through many areas in the system. The main consideration for optimization is in the backend on SQL Server and the disk subsystem to create as much Disk I/O parallelism as possible. An inadequate design in this area will have many downstream affects through the system.

The best practices cover the majority of capabilities used in the field and is not an exhaustive list of every feature. This document can be applied in many situations, and is meant to be a resource guide of recommendations for Project Server 2007.

4. Additional Factors

There are three main initiatives in the IT organization today adding a level of complexity to Project Server 2007 deployments.

- Consolidation
- Virtualization
- Outsourcing

Consolidation

In the context of this guide, consolidation refers to the co-existence with other Microsoft applications, namely SharePoint Server. The guidelines do not cover sizing or best practices in this scenario. Please read the EPM and Office SharePoint Server 2007 Coexistence — Intranet Scenario (http://www.microsoft.com/downloads/details.aspx?FamilyID=02ADC262-0E9D-4E8E-92B4-2E2BB64945B8&displaylang=en) whitepaper for prescriptive guidance.

Consolidation provides challenges in the Project Server 2007 implementation when different teams design independent solutions without collaborating on the specifications and deploy into a single system. Obviously there are cross-impacts and performance sizing does not take the accumulative throughput requirements for all the applications and as a result end up with an underperforming solution.

Virtualization

Virtualization initiatives have been growing stronger every year and is the accepted the platform of the future. A virtualized production Project Server 2007 platform needs to be carefully planned and designed to avoid potential performance related challenges.

1. SQL Server – At this time virtualizing the database server together with the disk subsystem has yielded less than desirable results. Later in this guide, the Object – CPU section explains that the limitation of the virtual CPU (Enterprise Edition only allows 4) undersizes the SQL Server CPU count. This together with an inefficient disk subsystem compounds the issue.

2. Virtual Hosts – Many organizations have their Project Server 2007 solution hosted in a virtual farm without understanding the specifications of the host container, how many other guests are running under the same host and the available system resources provided to the applications.

Outsourcing

Organizations using outsourcing often do not have the flexibility to create a custom designed solution or implement best practices specific to their application needs. The outsourced company has rigorous standards and change control processes that the organization must comply with may limit the level of agility, optimization and operational support.

2. How to Use This Document

This document assumes the audience is familiar with Project Server 2007 and its supporting technologies.

Refer to section 3 – "Container and Object Best Practices" for definitions on Containers and Objects. Each container has a relative object as described in the convention below.

Definition	*This section provides the definition of the object*
Guidelines for Acceptable Performance	*This section provides the best practices and supporting statements*
Calculation Factors	*This section provides the performance and scalability calculations to support the best practices*
Scope of Impact	*This section provides a list other objects impacted in the system when best practices are not implemented.*

3. Container and Object Best Practices

The following tables are categorized under three container areas which represent key objects focusing on recommended guidelines for acceptable performance. The guidelines are not hard limits and can be exceeded when taking the scope of impact into consideration based on feature usage within the solution. It is expected, when exceeding the guidelines, that there may be performance degradation and the tolerance for this would be solution and user specific.

The three containers are:

- **Platform**: Collection of application and system related objects
- **Data**: Collection of project, resource and security related objects
- **User**: Collection of user location, workload and user type related objects

1. Container: Platform

A. Object: Farm

Definition	A server farm represents the top-level element of a Project Server implementation. Individual server farms provide physical isolation from each other.
Guidelines for Acceptable Performance	Consideration needs to be given to the ratios of servers in the farm by server role. Typically SharePoint web fronts ends (WFE) should not exceed 5 servers in the farm, however, with the Project Server solution there is a balance between the WFE and application server before reaching diminishing returns. The SQL Server is farm agnostic and can have several farms leveraging its database capabilities together with the SQL Server shared across multiple applications such as Office SharePoint Server, Exchange Server and many others. Ensure each application has its own SQL Server Instance to manage SQL Server resources across each application. On a 64 bit platform, the ability to increase memory beyond the 2Gb limit provides a smaller physical footprint for servers and the number of processors becomes more of a factor than the number of servers. Performance degradation has been observed under the following: 1. Web Front End (WFE) more than 4 2. Application Servers more than 3 3. SQL Server – shared across multiple applications and contention across system resources 4. Total Queue Threads more than 12 for each queue type

Maintaining performance and optimizing one area of the farm requires an intricate balance all other server roles and they all need to be fine tuned in concert with each other.

An additional consideration needs to be given to a virtualized farm and how the physical requirements are translated into their virtualized logical representation. You must also ensure that adequate resources are available for all the virtual platform components.

In an extended farm scenario where Office SharePoint Server is also used for additional web applications, the system and application resources need to be dedicated to the farm for each feature being enabled. Refer to **EPM and Office SharePoint Server 2007 Coexistence — Intranet Scenario** http://www.microsoft.com/downloads/details.aspx?FamilyID=02adc262-0e9d-4e8e-92b4-2e2bb64945b8&displaylang=en for more details.

Ensure all the servers in the farm have synchronized system clocks. When a server has a system clock out of sync it will impact any timer jobs that need to run and queue jobs on the application servers may likely fail.

Dedicate a SQL Server to the Project Server 2007 farm and do not run any other farm roles on the server. Enable the SQL Server to keep pages in memory to prevent memory pressure and swapping memory to disk.

Enable Windows fast file initialization. When SQL Server creates or expands a file it must first be initialized. Fast file initialization is enabled at the OS level and grant user right “perform volume maintenance tasks” to the account under which SQL Server service is running.

Disable hyperthreading on the SQL Server as it can have a negative impact on processor loads.

Environments – A minimum of four environments are recommended to support a Project Server 2007 solution. Multiple instances can be provisioned on each Share Services Provider to accommodate training and isolated business unit needs.

1. Production
    a. Physical servers
2. Staging
    a. Pre-production environment mirroring production on a single capacity.
    b. Web front end and application server can be virtualized
    c. SQL Server can be virtualized using disk pass-through for all database files to physical disks.
3. Development

a. All server roles can be virtualized. Set guest image to fixed disk.

b. Create multiple instances of Project Web Access

4. Operations/Quality Assurance

a. All server roles can be virtualized. Set guest image to fixed disk.

b. Used for patch management, Cumulative Update and functional testing

Calculation Factors

Due to 64 bit architecture the calculation factors focus on system resources and not the number of servers.

Typical Ratios:

1. Web Front End CPU to Application Server CPU – 1:1
2. Web Front End CPU to SQL Server CPU – 2:1
3. Queue Threads – 2 per Application Server per CPU; not to exceed 12 threads for the entire farm.

Example – assuming 64bit platform:

	Scenario 1		Scenario 2		Scenario 3	
Resources	**CPU**	**RAM**	**CPU**	**RAM**	**CPU**	**RAM**
WFE	2	4	4	8	8	8
APP	2	4	4	8	8	8
SQL	4	8	8	16	16	32
# Q Threads	4		8		12	

Scope of Impact

- **32bit/64bit** – ensure all servers are either 32 or 64 bit. Do not mix the server versions as the farm may likely behave erratically across all the application requests.
- **SQL Server** – Multiple farms dedicated to the same SQL Server (or other applications) will degrade performance if the database is not sized appropriately for all farms. (Black box concept)
- **Virtualization** – Physical Host machine can dedicate required logical resources across guest machines by role and ratios.
- **Office SharePoint Server Co-existence** – Careful design is required to allocate required resources across varying applications sharing the hardware, the Shared Services Provider and data sharing capabilities.

B. Object: Shared Services Provider

Definition	A Shared Services Provider (SSP) provides a common set of services and service data to a logical grouping of Web applications and their associated sites.
Guidelines for Acceptable Performance	SSPs are associated with Web applications. All sites within a Web application are served by the SSP that is assigned to the Web application. An SSP can host services for multiple Web applications. For additional detailed information about SSP review "Logical architecture components" at: http://technet.microsoft.com/en-us/library/cc263121.aspx#section2. Separate SSPs provide process isolation of content, profiles and search results. Process isolation is accomplished in the following ways: • Each SSP resides in a separate IIS application pool • Each SSP uses a unique set of service accounts to run the services provided by the SSP, such as search and content crawling. For details on the SSP review **EPM and Office SharePoint Server 2007 Coexistence — Intranet Scenario** http://www.microsoft.com/downloads/details.aspx?FamilyID=02adc262-0e9d-4e8e-92b4-2e2bb64945b8&displaylang=en It is possible to create multiple instances of SSPs per farm dependant on the available system resources, Office SharePoint Server specifies no more than 20 SSPs per farm, however, for Project Server 2007 it is recommended to have no more than 3. Ensure you understand all the implications of the decision to provide separate SSPs with separate PWA instances for separate business units. Usually, there is a correlation between the number of SSPs and the number of PWA instances exists. It is possible to create a certain number of PWA instances per each SSP dependant on the available system resources. However, for Project Server 2007 it is recommended not to have more than 10 PWA instances per SSP. Increase the timeout setting for the SSP controlling access to the Project Server databases http://blogs.msdn.com/brismith/archive/2009/07/31/project-server-2007-timeout-settings.aspx . By default it is 30 seconds, consider setting it to 90 seconds.
Calculation Factors	Improvements to the SSP are addressed on the 64bit platform by increasing the memory allocation to the web application and application pool. Ensure Project Server 2007 and all its prerequisite operating system and database software have installation options for 64-bit computers. For detailed info about benefits the of 64 bit hardware and software review:

	• **Advantages of 64-bit hardware and software (Project Server 2007)** - http://technet.microsoft.com/en-us/library/dd745012.aspx . • **Advantages of 64-bit hardware and software (Office SharePoint Server 2007)** - http://technet.microsoft.com/en-us/library/dd630764.aspx • **Why WSS 3.0 x64 and OFFICE SHAREPOINT SERVER 2007 x64... What's the deal** - http://blogs.msdn.com/joelo/archive/2007/01/12/why-wss-3-0-x64-and-moss-2007-x64-what-s-the-deal.aspx.
Scope of Impact	• **System Resources –** Additional SSPs will cause degradation to performance. • **32bit/64bit –** 32bit platforms will be more affected by this limitation. On 64bit platforms ensure that you configure resource limits and align them with the available system resources. • **Data Isolation –** Creating separate SSPs introduces a level of data isolation. Ensure the imposed security boundaries align with the business units and user requirements.

C. Object: Application Pool

Definition	An application pool is a grouping of one or more URLs (or Web sites as they are represented in IIS) served by a worker process. Each application pool has its own worker process and identity (security account) which prevents two processes from interacting.
Guidelines for Acceptable Performance	Separate Internet Information Services (IIS) application pools are typically implemented to achieve process isolation between content. Application pools provide a way for multiple sites to run on the same server yet still have their own worker processes and identity. The guideline for acceptable performance is a maximum of 8 application pools per web server. Refer to - http://blogs.msdn.com/joelo/archive/2007/10/29/sharepoint-app-pool-settings.aspx for details. Ensure the “Max Used Memory” setting utilizes all the available RAM in the Web front-ends (WFEs) Increase the “ASP.Net:” setting for the Project Server Web Site to reduce the PWA page and view timeouts. http://blogs.msdn.com/brismith/archive/2009/07/31/project-server-2007-timeout-settings.aspx Project Sever 2007 requires that application pools be recycled regularly. Follow

	these recommendations to keep sites up and running: 1. Turn off the recycle worker process 2. Configure a virtual memory-based recycle to occur at 1700 MB. 3. Use time-based recycles for certain times in the day to avoid running during a heavy load period. Set a scheduled recycle ~30 minutes before peak traffic begins 4. Recycle application pools at different times for different web servers. 5. Recycle application pools at different times for different IIS web sites
Calculation Factors	When each Web application starts, it loads an App. Domain into a memory. (.NET framework instance) Depending on the complexity of all active Web applications, memory required to run all desired Web applications may grow significantly and general guidance for the maximum number of applications sharing the same application pool may not apply. The memory overhead of an application pool is 100-450 megabytes (MB) plus any memory for the applications running in the application pool process space. The limit for the number of application pools is influenced by the available memory on the system. That is, the number of application pools is dictated by the following two factors: • Available addressable memory. • The size of the application running in the application pool. A typical farm has 4 applications by default. At ~ 450Mb each (under certain conditions) the memory footprint can easily attain 1.5Gb. On a 32bit platform this would introduce performance degradation.
Scope of Impact	• **System Resources** – Additional Application Pools will consume system memory. Certain operations such as the Queue processing and RDSSync would be negatively impacted. • **32bit/64bit** – 32bit platforms will be more affected by this limitation. On 64bit platforms ensure that you set governance to limits that will be created and align them with the available system resources. • **Virtualization** – The number of guests running in a host and the allocation of logical memory for all the application pools running in the farm.

D. Object: Web Application

Definition	A Web application is an IIS Web site that is created and used by SharePoint Products and Technologies. Each Web application is represented by a different Web site in IIS.
Guidelines for Acceptable	The guideline for acceptable performance is to implement 10 or fewer Web

Performance	applications per each SSP and map directly to the number of instances created. For additional detailed guidance, review Logical architecture components (Windows SharePoint Services) at http://technet.microsoft.com/en-us/library/cc287815.aspx#section4 and Plan for Software Boundaries at: http://technet.microsoft.com/en-us/library/cc262787.aspx. PWA instance(s) should use its own web application that is separate from web application hosting other services. Increase the "Connection timeout:" setting for the Project Server Web Site to reduce the PWA page and view timeouts. http://blogs.msdn.com/brismith/archive/2009/07/31/project-server-2007-timeout-settings.aspx Don't use web gardens as applications using multiple worker processes may have a negative impact on performance.
Calculation Factors	You can create up to 99 web applications per SSP. It is important to note that each ASP.NET page generates a separate dynamic-link library (DLL) for each Web application. The separate DLLs consume memory, limiting the number of Web applications that can run on a server. When each Web application starts, it loads an App. Domain into a memory. (.net framework instance) Depending on the complexity of all active Web applications, memory required to run all desired Web applications may grow significantly and general guidance for the maximum number of applications sharing the same application pool may not apply.
Scope of Impact	• **Project Web Access** – The number of users on each instance of PWA and application processing for each one will degrade performance as the number of web applications increases. • **32bit/64bit** – 32bit platforms will be more affected by this limitation. On 64bit platforms ensure that you set governance to limits that will be created and align them with the available system resources. • **Virtualization** – The number of web applications running per guest on each host and the allocation of logical memory for all the web applications running in the farm.

E. Object: Database

Definition	The Project Server Farm uses SQL Server and a number of databases to support the solution.
Guidelines for Acceptable Performance	By default a single Project Server farm creates 9 databases. Each additional instance creates 4 new databases and possibly a new content database (if

configured to recommendations for hosting the instance workspaces). The recommendation to limit the number of instances per farm to 10 will create a total of 54 databases. The database administrator needs to be aware of this and allocate operational policies to the backup and restore process, service Line agreements, disaster recovery, mirroring and IT cost to manage the process.

Defragment your physical volumes on a regular schedule for increased performance. LUNs need to be 20%–50% larger than the data stored on them to allow for effective defragmentation of the data files.

The Temporary Database (TempDB) is an important and overlooked database in the farm. It is primarily responsible for performing a lot of the calculations, security trimming and management of Queue operations. Optimizing the disk subsystem and file allocation will greatly improve performance, even for the smaller deployments. TempDB is the most active database in the farm.

Schedule backups for minimal disruption at times when the system is not under a heavy load.

Ensure the SQL Agent Job is started which cleans up expired user sessions in the Shared Services Provider Database.

Project Server introduced complex data structures in Project Server 2007 to manage a federated data set. These show up primarily as GUIDs in the table and in additional memory documents for multi-level undo in Project Professional. The major advantage of federation is the GUIDs are unique across all space and time. This is helpful if there is consolidated data from multiple SQL Servers into one table, as in a data warehousing situation. The main disadvantages to using GUIDs as key values are the slower performance, introduced fragmentation, and their size. Due to the random nature of unique identifier values, insert performance degrades when you have a large table with an indexed unique identifier column. It's important to implement a SQL Server maintenance plan checking the following tasks:

- Backup
- Check DB Integrity
- Rebuild Index
- History Cleanup
- Maintenance Cleanup

Refer to Database maintenance for Office SharePoint Server 2007 (white paper) http://technet.microsoft.com/en-us/library/cc262731.aspx, How to defragment Windows SharePoint Services 3.0 databases and SharePoint Server 2007 databases http://support.microsoft.com/kb/943345 and http://blogs.msdn.com/chrisfie/archive/2007/10/25/epm-database-maintenance-

	tasks.aspx for more details. **Project Server and Temp Databases** • Data files should be spread across unique LUNs • If there are limited LUNs then place Draft and Published database files on separate LUNs • Log files separated to unique LUNs • Data and Log files pre-sized and set to fixed amount • TempDB Data and Log Files pre-sized and set to fixed amount • RAID 10 is the best choice when cost is not a concern and highly recommendations for intensive write operations on TempDB and Log files • RAID 5 will be sufficient under smaller application load and acceptable on data files for the content which has a mix of write and read operations • 15K RPM drives • Do not create secondary file groups for secondary files (NDF) on a different drive as this is an UNSUPPORTED situation. As a result, Central Admin backup and restore may fail
Calculation Factors	**Database Sizes** • Typically the Archive DB and Published DB are double the size of the Draft DB – 1:2 ratio. • The Reporting DB is typically three times the size of the Draft DB – 1:3 ratio. • To determine the approximate size of data in the farm for storage use the Data Population tool on a sample Draft DB http://www.codeplex.com/EPMDatapop and the ratio above to extrapolate total data size for an instance. Applying actuals over a period of time will significantly increase the database size **TempDB** • Crate 1 TempDB file per SQL Server CPU • All files must be the same size • Spread data files across unique LUN (separate to other databases) • Separate Log file to unique LUN (separate to other databases) • Optimal TempDB data file sizes calculation [MAX DB SIZE (KB)] X [.25] / [# CORES] = DATA FILE SIZE (KB). The result (starting size) should be roughly equal to 25% of the Reporting DB size for each instance in the SSP. **Index Fragmentation and Index Rebuild**

	• Defrag/Rebuild index if fragmentation level is above 25% • The index type is not a heap (meaning not indexed and cannot be defragmented unless the table is truncated and data re-added) • The page count is greater than 1000. A page count smaller than 1000 represents only a fairly small amount of data and isn't severely impacted by fragmentation. However as the size of the page count increases along with fragmentation levels that is where performance will start to degrade. • Consider defragmenting on a regular basis especially if there are a large number of updates. E.g. Schedule it such that the defragmentation occurs after hours after the business "update' days.
Scope of Impact	• **Views** – Retrieval of view data will be slow on data with fragmented indexes and may cause view timeouts. See the SSP, Web Application and Application Pool objects for reference to increasing timeout values. • **PWA** – Page loads may be slow due to an overtaxed TempDB that is not able to perform the security trimming of user authorization data • **CRUD** – This acronym represents long create, read, update and delete operations on data in fragmented indexes experienced by Project Professional users and queue operations. • **Disk Files** – When not using RAID 10 or unique spindles for the Draft and Published databases the queue tables and queue operations may likely degrade in performance. • **SAN** – The shared/dedicated scenario of placing files on a SAN without knowing the spindle allocation to the LUNs and spindle sharing across volumes on other active applications. • **Virtualization** – Although supported it is not recommended to have Project Server databases running on a Virtualized Host SQL Server. For smaller environments, where testing has been done, it is deemed acceptable to virtualize the SQL Server ensure the database files reside on physical disks on a SAN or using disk pass-through to physical attached storage.

F. Object: System - Disk

Definition	A hard disk drive is a non-volatile storage device that stores digitally encoded data, operating system files and software titles on rapidly rotating platters with magnetic surfaces. It is also used to perform temporary data operations and for temporary memory storage through a page file on disk.
Guidelines for Acceptable	Disk topology and design is a priority in a Project Server 2007 solution. In fact, the SQL Server and disk design is where the majority of performance gains are made.

Performance	Due to the quantity of data moved around the system, application calculations and data read/writes across as few as 9 databases, optimizing Disk I/O is of paramount importance. Use Diskpart to align the LUNs. Do not exceed 80% utilization on LUNs. Set NTFS allocation unit to 64 kb. **Disk Topology** • Direct Attached (DAS) - Direct attached disks often have higher maximum bandwidth. This can be very important for bottlenecked workloads. The disk sizing and RAID selection would determine the number of DAS drive bays is required on each server. Typically we see Web Front End and Application Servers using DAS. This also provides flexibility for IT to manage and design a solution specific to Project Server. • Storage Area Network (SAN) - SAN is a centralized network of storage devices. A centralized storage network allows for server power and utilization to focus on the business application and I/O processing to be offloaded. A SAN does not provide flexibility for IT to manage and design a solution specific to Project Server. **RAID** • Choosing the correct RAID level for your solution is a key design decision. RAID systems are designed for reliability through redundancy in disk arrays. RAID1, RAID5, and RAID10, provide protection against disk failures. Only RAID1 and RAID10 protect against multi-disk failures, RAID5 only protects against a single-disk failure. RAID0 provides no protection against disk failures because it only performs data striping. • RAID5 has a much lower write performance than any other configuration because it requires extra reading and writing activities for the parity blocks in addition to reading and writing the data. Project Server 2007 databases are write intensive. • RAID10 yields excellent read-write performance and is recommended for Project Server 2007. • Increase the number of spindles per LUN to optimize I/O parallelism **Rotational Speed** • The rotational speed is the speed at which the disk spins. The higher the rotational speed, the more data the drive can read/write in a fixed time. • Rotational speeds at 15,000 rpm or faster are recommended for Project Server 2007. **Maintenance - Physical Volume File Fragmentation** • Defragment physical volumes on a regular schedule for increased performance!

	• LUNs need to be 20–50% larger than the data stored on them to allow for effective defragmentation of the data files **Disk Usage** • OS/Base Software. Use the C:\ drive to store the operating system and any other tool binaries and library files. • Log/Storage files. Use D:\ and E:\ drives for additional storage and capacity for ULS, IIS, EVT and BLG files. **Virtualization** • If the Web Front End and Application tier is virtualized consider the write activities of the ULS and IIS logs for Disk I/O contention. It is not recommended to have SQL Server virtualized in a production environment as noted in the "Object for System – CPU: section. In a staging or pre-production environment a virtualized SQL Server is acceptable. However, be sure to use Disk Pass Through for all the database data and log files allocated on physical disks.
Calculation Factors	Refer to the **Sizing Calculation Factors** section at the end of this document.
Scope of Impact	• Disk I/O contention • Slow Queue operations • Views timeout • Shared Services Provider timeout

G. Object: System - Network

Definition	The network interface card, more commonly referred to as a NIC, is the device that allows computers to communicate with each other
Guidelines for Acceptable Performance	Client connectivity through Project Professional using the Active Cache alleviates network contention; however, ensuring project managers maintain efficient schedules which reflect a 20% data change for each save. Servers in the farm, due to access to the Root Site for authentication, Content Databases for content and Shared Service Database for session management, are essentially two tiers. • Create a network segment from the Web Front Ends directly to the SQL Server. • Create a network segment from the Web Front Ends directly to the client access • Create a network segment from the Web Front Ends directly to the Application Servers • Create a network segment from the Application servers directly to the SQL

	Server. • Create a network segment for data backups If you are using a virtualized environment ensure the host network adaptors have enough bandwidth for all the guest machines Use Gigabit Ethernet switches with VLAN capabilities Use CAT6 cables for Gbe connectivity Set network speed and duplexing to 1Gb/s and FULL duplex mode.
Calculation Factors	Refer to the **Sizing Calculation Factors** section at the end of this document.
Scope of Impact	• Site-to-Site connectivity may impact performance and a hosted Terminal Services solution should be implemented to support these users • Web Front End timeouts may occur when the SQL Server access is not optimal for handling user requests to the databases

H. Object: System - Memory

Definition	RAM (random access memory) is the place in a computer where the operating system, application programs and data in current use are kept so that it can be quickly reached by the computer's processor.
Guidelines for Acceptable Performance	Using the 32bit platform requires optimizing virtual memory in IIS and the Application Pool. The 64bit platform is recommended for Project Server due to the quantity of core data and meta data moved around in the system. The design of the schedules and scalar depth has an impact on the sizing of the solution. Refer to the **PROJECT SYSTEM HEALTH REPORT V2.0** query in the appendix to monitor threshold values across the Project Server system.
Calculation Factors	Refer to the **Sizing Calculation Factors** section at the end of this document.
Scope of Impact	• Web Front End requests for task and timesheet submissions • Queue operations on the Application Server • SQL Server operations

I. Object: System - CPU

Definition	CPU stands for “Central Processing Unit”, and may sometimes be referred to by the term “processor”. A CPU consists of one or more electronic components that control the operation of a computer by following instructions and manipulating

	data.
Guidelines for Acceptable Performance	Typically there is a ratio to be observed between the tiers in the farm to ensure, for example, that Web Front Ends do not make more requests to SQL Server than the CPU can process. • Web Front End CPU to Application Server CPU ratio – 1:1 • Web Front End CPU to SQL Server CPU ratio – 2:1 The design of the schedules and scalar depth has a varying effect on CPU across the servers. For example, project schedules that increased the number assignments from 3 per task to 5 yielded a 40% spike in the Web Front End CPU to process the additional data. In a virtualized environment the CPU allocation is critical to the design. Logical processors do not have a 1:1 processing power as their Physical counterparts together with the host overhead. In SQL Server testing there was no linear scale across virtual processors and the net result showed only 75% gains for each virtual processor. In theory, a physical SQL Server with 4 CPUs when virtualized will provide only 3 CPU throughput. In Hyper-V the maximum logical CPU allocation is 4 and most SQL Server deployments require more than 4 CPUs. This is the main reason not to virtualize SQL Server in a production environment In Web Front End testing there was no linear scale across virtual processors and the net result showed only 75% gains for each virtual processor. In theory, a physical Web Front End server with 4 CPUs when virtualized will provide only 3 CPU horsepower. In Hyper-V the maximum logical CPU allocation is 4. When the host machine has enough physical resources to allocate across its guest machines it is possible to virtualize the Web Front End and Application tiers where the total CPU count required is not exceeded.
Calculation Factors	Refer to the **Sizing Calculation Factors** section at the end of this document.
Scope of Impact	• Web Front End timeouts against user requests and status submission • Queue operations • SQL Operations

J. Object: Logs

Definition	Logs are computer files with chronological recording of diagnostic data. In the Project Server 2007 system the main log files to consider are ULS (Unified Logging Service), IIS (Internet Information Services) and Events.

Guidelines for Acceptable Performance	Ensure each server in the farm has an additional drive other than the OS files with enough storage to manage all log files. Troubleshooting with verbose logging may temporarily require additional storage space; if that space is not allocated it may bring the server down. Consider using SAN storage external to each server when local drive space is not available Place logs on non-system partition Use event throttling to manage ULS log growth by selecting on critical events are logs.
Calculation Factors	Local storage on each server
Scope of Impact	• Disk I/O bound • Local storage

K. Object: Performance Counters

Definition	Performance Counters are used to provide information as to how well the operating system or an application, service, or driver is performing. The counter data can help determine system bottlenecks and fine-tune system and application performance.
Guidelines for Acceptable Performance	This section covers the Project Server counters only. The Office SharePoint Server and SQL Server TechNet sites cover the typical farm counters that should be monitored. For a full set of all the counters to monitor and a tool to help analyze the results refer to the Performance Analysis of Logs (PAL) Tool which has a Project Server file definition included - http://pal.codeplex.com/Release/ProjectReleases.aspx?ReleaseId=28708 . **Monitoring time-periods** Typically, based on the various uses of the performance data logs, there are three classifications of time periods: 1. Daily: This type of data is very useful in performance monitoring and troubleshooting because this is granular enough to give an idea about what areas to explore to pinpoint any problems. Depending upon what problems are identified, it is easy to analyze just a few hours of data to get an idea of the loads on the system, memory state, CPU processing time, etc. 2. Monthly: This type of data is typically useful in database maintenance issues such as index tuning, re-building the indexes or creating back-up plans. This data is not granular enough for troubleshooting and other

performance optimizing diagnosis.

3. Quarterly: This type of data is typically useful for trend analysis. The data at a quarterly level will help in predicting future requirements based on growth of the database and requests to the database. This is also a good indicator for preparing for high/low demand seasons for example, a database serving an online-retail store during holidays.

Data points

As a general rule of thumb, a 3-second counter delay is typical in tests that run 0.5 hours to 4 hours. Longer tests should adjust sampling intervals to 10-15 seconds. Shorter tests may require 1-second intervals.

Project Server Counters

The generic WFE and SQL counters that can be monitored come with thresholds and levels of potential alerts. Project Server does not provide these thresholds as the counter values require cross referencing with related counters to determine the health of the system. The counter name is self explanatory. The reference column provides details to the counter relationships when analyzing the reports.

The process for analyzing the data and building a health chart for the farm follows a flow of 6 main categories the counters fall into.

Category 1: How busy is the application server? – How many PSI Calls/sec and how the server resources are handling the load.

Category 2: How many jobs are arriving (Enqueued)? – Rate of jobs arriving in the Queue

Category 3: Job wait-time/starvation (Queue Depth)? – How long a job sits in the queue before being dequeued

Category 4: How active is the processing (Queue Threads)? – How many threads are available to process and how many are being used?

Category 5: How long are wait times/job starvation? – How many jobs are starved in the queue without threads or long running jobs

Category 6: How many jobs are going out (Dequeued)? – Rate at which jobs are processed

The reference column below has the **category number** the counter belongs to as categorized above and highlighted for analysis. Counters with the **category** association are key counters to monitor for Project Server health.

ProjectServer:User Activity	Reference
PSI Calls per Second	**Category 1**

	This counter illustrates how busy the application server is. Examining which PSI calls are being made shows which web services are busiest for the source requests. For example: on a Friday timesheet/status updates from Project Web Access or Monday schedule updates from Project Professional. To analyze this counter it should be correlated by PSI calls (use the IIS Log report to trace PSI usage patterns) and cross referenced with the server CPU and RAM to determine a threshold for the farm.
ProjectServer:QueueGeneral	**Reference**
New Jobs / Minute	**Category 2** This counter shows the rate of new jobs arriving (enqueued) to the queue. To analyze this counter you should correlate it with the **Category 1** counter (how busy the PSI is) and **Category 3** counter (queue depth) to determine if the queue can keep up with the load. This is a typical scenario during peak usage on the system.
Average Unprocessed Jobs / Day	**Category 3** Count of all unprocessed jobs, averaged for the last day and the count of all unprocessed jobs for the moment. These counters show queue depth and the amount of jobs waiting in the queue. To analyze this counter it should be correlated with **Category 2** counter to determine how many jobs are being Enqueued before being processed.
Current Unprocessed Jobs	
Active Job Processing Threads	**Category 4** The number of threads is processing jobs at any given moment. To analyze this counter it should be correlated with the queue setting thread count. If the counter value equals the fixed value set in the queue for a period of time there is thread starvation. This could

	mean increasing the thread count value or increasing the CPU power of the Application Server.
% SQL Retries / Day % SQL Retries / Hour	Whenever a SQL Command as part of the queue fails, it is executed again, up to a pre-specified value set in the queue management page. The counters correlate with the **SQL Call / (x)** counters showing the workload on the server.
SQL Calls / Hour/Day SQL Calls / Minute SQL Retries / Minute	These counters show contention and workload on the SQL Server. To analyze these counters they should be correlated with the Disk IO on the server to determine bottlenecks and threshold values. The quantity of calls relates directly to the size of the jobs in the queue and the size of the projects. Larger jobs correlate to larger datasets and increase the # SQL calls.
ProjectServer:QueueJobs	**Reference**
% Jobs Failed / Day % Jobs Failed / Hour % Jobs Retried / Day % Jobs Retried / Hour	% of jobs failed previous hour and 24hrs % of jobs retried previous hour and 24hrs These values are calculated from the Averages counters below
Average Processing Time / Day Average Processing Time / Minute	Average job processing time
Average Wait Time / Day Average Wait Time / Minute	**Category 5** The counters show the wait times, queue depth and starvation in the queue to process waiting jobs. This can also be caused by lack of threads available or long running jobs. To analyze these counters they should be correlated with the **% SQL Retries**, **SQL Calls /(x)** and **Category 4** counters. Remember that one queue thread = one job

		(project). If you have a series of large projects that begin processing in the queue at the same time other jobs (projects) in the queue will back up until the large projects have completed. Smaller jobs do not get processed in a priority. Although the queue and PSI is a multithreaded application, the sequence of processing jobs on a queue thread is synchronous for that specific job. Another example is if you have a large project in the queue by itself and 4 available threads – this will not process any quicker – as the job will run on the same queue thread synchronously. Therefore do not expect large jobs to run any quicker through the queue when there is no significant queue depth.
	Jobs Failed / Minute	**Category 6** These counters show the rate at which jobs are processed in the queue (Dequeueing). To analyze these counters they should be correlated with **Category 2**, **Category 3 and Category 4** counters. The balance to strike is between the number of new jobs arriving, the threads available to process and the # of unprocessed jobs. Monitoring this group of counters will determine the bottleneck and where optimization needs to take place.
	Jobs Processed / Hour/Day	
	Jobs Processed / Minute	
	Jobs Retried / Minute	
Calculation Factors	As mentioned previously, there are no out of the box thresholds that can be set for Project Server counters as the farm configuration, system resources, disk subsystem, usage patterns and data profile are variables that require correlation through analysis of the system performance. The correlated values can then be set as thresholds for the specific farm.	
Scope of Impact	Setting the best practice of monitoring theses counters with regular analysis will help trend the system and identify bottlenecks from the hardware to the project dataset size and usage patterns.	

2. Container: Data

The Data Container references project scalar depth highlighting key objects that provide cross impacts to scalability and performance at a user and technical level. Functional design of a Project Server solution needs to be orchestrated within the technical capabilities spectrum to support the desired throughput of the data. Scalar Depth is the key design attribute in the performance and scalability context with functional usage as it introduces additional calculation overhead within the application, security trimming of permissions and data inserts/reads. These factors quantify the perceived technical impacts across the solution. The appendix contains the PROJECT SYSTEM HEALTH REPORT V2.0 query which an administrator can run on a regular basis to determine threshold values from the Scalar Depth across the Project Server solution.

A. Object: Projects

Definition	Microsoft Project Professional uses the Critical Path Method (CPM) of scheduling. Project Schedules consist of activities or tasks and their relationships through dependencies and constraints. CPM is a method of calculating the total duration of a project based on a specified project start date and on the individual duration of tasks and their dependencies. Master Projects can be used to provide rolled-up reports across multiple projects and also create external dependencies between tasks in different projects.
Guidelines for Acceptable Performance	There are published guidelines found at Plan for software boundaries (Project Server) http://technet.microsoft.com/en-us/library/cc197693.aspx. These apply more to the client than to the server but there are cross dependencies with the server that also need to be taken into consideration. The other data objects discussed within this container are relative to the performance of projects in the system and the variety of profiles may change the guidelines. For example: Having a solution with 5000 projects and an average of 3 tasks / project may display similar performance characteristics when compared to a solution with 20 projects that have an average of 750 tasks/project. Increasing the number of assignments, baselines and custom field values provides a varying response time to certain operations. Asking the question **"how many projects can Project Server manage?"** is dependent on the relationship of other core/meta data. Another example is how many sub-projects can be opened simultaneously from the Server? There is no specific number as the dependency is a memory limit on the client machine running Project Professional. Project Professional is 32 bit application, if the client machine has 4Gb RAM and no other applications running it may effectively use 2Gb of virtual memory. This memory footprint determines the number of projects. Assuming the projects are 20mb each may allow ~100 projects opened within a Master Project, however, assuming projects are 120Mb

each may only allow ~17 projects opened within a Master Project.

The concept of master projects and subprojects is important in Microsoft Project 2000 and earlier because this is the best way to organize, work with, and track the progress of multiple projects in these versions. Beginning with Microsoft Project Server 2007, the need for master projects is reduced. Resource management across multiple projects is simplified by the use of a single global resource pool, and the relationships between projects can be managed by using enterprise outline codes and custom fields. In Project Server 2007 the saving of master projects to the server is fully supported and recommended. In previous versions this was not a recommended approach. Considerations and Best Practice Methods:

1. Store each individual project in Microsoft Project Server. Each subproject file must be stored in the database so that you can view the summary task for each subproject and access the project view. If you publish only the master project file, the only tasks that are displayed in a project view for the master project file are those tasks in the master project file
2. First create the master project and save to the server then insert all sub projects
3. All project files must reside in the Microsoft Project Server database. This means that you must import and publish existing files to Microsoft Project Server. When you import a master project, the subproject links are not automatically updated. You must open the master project and update the links to use the correct location to the subproject plans. This ensures that the correct subproject file is updated when new information is added to the subproject from within the master file.
4. Create a custom field to track the relationship between master project files and subproject files in Microsoft Project Web Access. In Microsoft Project Web Access, there are no indicators or fields to distinguish whether a project is a master project or a subproject file. To identify a project as a master, a subproject, or a standalone project, create a custom Enterprise Project Text field or Enterprise Project Flag field. You can use this field to group, filter, or create new views to organize and view master files and subproject files.
5. To edit sub projects always open the master project first and then open sub projects in read/write mode and make any edits directly in the sub projects
6. Do not open multiple windows of the same project. For example: Master project and sub projects in read only mode and then opening the sub projects in write mode. This scenario may likely cause a document conflict in the active cache mechanism and/or cause the application to stop responding.

7. Consider turning off auto calculations as this may take considerable time across all open projects for each change made to a plan. Complete all changes first then manually calculate the changes in a single operation.

When determining guidelines one should consider all the factors and quantify the result of the data profile and scalar depth relative to the usage of the solution.

Ensure there is a regular database maintenance plan running on the SQL server to provide faster insert and read operations. Fragmented indexes slow down the server performance retrieving the views for the users and queue operations in the back end. Refer to http://blogs.msdn.com/chrisfie/archive/2007/10/25/epm-database-maintenance-tasks.aspx for more details.

Limit the number of enterprise calendars as this creates calculation overhead through the precedence of the base calendar, the project calendar, the resource calendar and task calendar.

Calculation Factors

There is no theoretical limit for the total number of projects. The factors are determined more by the project meta data for scale. For large customer's usage we find the total count up to 15,000 per instance and the active project list is governed by fiscal policy. Therefore with the appropriate archival and reporting process in place a typical average we find in the field is ~2000 projects within a fiscal period. Projects of this quantity usually have short durations of 3 – 6 months. For other large customer usage we find only 15 – 20 projects, however, they are multiyear durations with large number of tasks. Customers with this type of profile usually require a more optimal technical and functional design to achieve performance than a profile of 2000 projects with 3 month durations.

The table below represents key numbers to consider for scalar depth – the numbers presented are **not limits** – they **are thresholds** and represent the usage profile for Project Server that yields acceptable performance. There are many systems that go beyond this threshold where additional optimization, functional design and process review are performed to maintain the desired throughput.

Object	Threshold
Active projects	**10,000**
Project duration	**24 months**
Number of inserted subprojects	**16**
Number of project custom fields	**30**
Total Enterprise custom field values	**32,000**
Number of baselines	**7**
Number of calendars	**5**

† The appropriate architecture and physical design is required to support the thresholds noted in the table above.

	When using Project Server with a different set of threshold values the impact can be observed throughout the solution from the client machine, system resources on the web front end, application server and SQL Server. For example in one of the performance testing labs we noted the web front end CPU required an additional 25% when increasing the number of assignments per task from 3 to 5. Refer to the **PROJECT SYSTEM HEALTH REPORT V2.0** query in the appendix to monitor threshold values across the Project Server system.
Scope of Impact	• Project Professional – longer than expected save times • Application Server sizing: Queue Threads – The total number of projects saved/published are associated to a thread. One project = one job on a thread (refer to the User Container for more detail on threads). The maximum number of threads per farm is 12. The more projects saved during peak time, the greater wait times for processing. Projects that have a lot of task data and custom field values will take longer to process. It may look like the queue is not operating. However, the long running jobs are taking up the queue threads. • Storage – determining the maximum disk space required for storage across the 4 databases • SQL Table Index fragmentation

B. Object: Tasks

Definition	Tasks describe project work in terms of sequence, duration, and resource requirements. There are two levels of tasks: • Summary tasks, which summarize or "roll up" the durations and costs of tasks. A summary task behaves differently from other tasks. You can't edit its duration, start date, or other calculated values directly because this information is derived or "rolled up" from the detail tasks. • Subtasks that together make up the significant events in the life of a project.
Guidelines for Acceptable Performance	Although there are published guidelines found at Plan for software boundaries (Project Server) http://technet.microsoft.com/en-us/library/cc197693.aspx these apply more to the client side boundaries. Cross dependencies with the server side boundaries that need to be taken into consideration exist as well. The average task usage seen in the field is less than 1000 per project. However, there are always strategic projects where the task count is typically between 1000

	and 5000. Certain vertical industries like construction, government and aerospace have fewer projects in the system but the average task count per project can be in the 10,000 – 100,000 range. Projects with this number of tasks, together with associated custom field values present a large data set for the Web Front End to handle and the backend to process. To achieve an acceptable performance, limit the maximum number of tasks per project to 5,000 together with diligent use of task custom fields. Task duration also increases the total project duration, timephased data and scalar depth which grow exponentially with assignments. Schedule best practice recommends tasks have an average duration of 8 – 80 hours or 1 – 10% of the total project duration. Consideration needs to be given to the task dependencies and relationships through the task dependency network as these cause calculation overhead. Users who maintain optimized schedules and follow project management best practices typically do not experience performance issues in Project Server. For example, referencing the table below with the noted threshold for **Percentage of schedule change** shows a 20% change, meaning 80% of task data should not be affected by any schedule updates for that status period. We find that project managers experiencing performance issues are using volatile custom fields which "dirty" all the schedule data and/or the dependencies through the network changes all the task values. This effectively means the entire plan has to be saved, minimizing the effectiveness of the client side caching, sending a large dataset through the system with each save and requiring additional backend processing. Most customers experiencing performance related issues show a data profile with a high degree of data skew across projects relative to task and assignment customer field values. Typically a few projects have a high number of values and the majority of them have very few. This data skew can impact the query plan logic. It may be required to create custom Plan Guides for your solution - http://technet.microsoft.com/en-us/library/ms190417.aspx. Ensure there is a regular database maintenance plan running on the SQL server to provide faster insert and read operations. Fragmented indexes slow down the server performance retrieving the views for the users and queue operations in the back end. Refer to http://blogs.msdn.com/chrisfie/archive/2007/10/25/epm-database-maintenance-tasks.aspx for more details.
Calculation Factors	The table below represents key numbers to consider for scalar depth – the numbers presented are **not limits** – they **are thresholds** and represent the majority of the usage profile for Project Server that yields acceptable performance. There are many users who go beyond this threshold where additional optimization, functional design and process review is performed to maintain the desired throughput.

Object	Threshold
Number of tasks	**5,000 per project**
Task duration	**24 months**
Number of task custom fields	**30**
Percentage of task schedule change	**20%**

† The appropriate architecture and physical design is required to support the thresholds noted in the table above.

When using Project Server with a different set of project/object threshold values the impact can be observed throughout the solution from the client machine, system resources on the web front end, application server and SQL Server. For example in one of the performance testing labs we noted the web front end CPU required an additional 25% when increasing the number of assignments per task from 3 to 5.

Refer to the **PROJECT SYSTEM HEALTH REPORT V2.0** query in the appendix to monitor threshold values across the Project Server system.

Scope of Impact

- Client machine – additional RAM and CPU to manage larger project size and CPM calculations
- Active Cache – volatile schedules creating large data sets
- Web Front End – Additional CPU and RAM to manage node consistency scheduling across large data sets
- Application Server – Additional CPU and RAM to manage large data sets and queue operations
- SQL Server – Additional system resources and disk optimization to manage large data sets and queue operations
- The number of custom field values
- Views – Number of tasks per view and custom fields
- SQL Table Index fragmentation

C. Object: Assignments

Definition

Assignments are the associations between specific tasks and the resources needed to complete them. More than one resource can be assigned to a task. Generic as well as individual resources, work resources, material and cost resources can be assigned to tasks.

Guidelines for Acceptable Performance	Although there are published guidelines found at Plan for software boundaries (Project Server) http://technet.microsoft.com/en-us/library/cc197693.aspx these apply more to the client side boundaries. Cross dependencies with the server side boundaries that need to be taken into consideration exist as well. Caution should be exercised when assigning resources to summary tasks. Although there may be valid scenarios where this is desirable - If you assign a resource full time to a summary task, don't also assign the same resource full time to subtasks under that summary task, or you may cause unwanted resource over-allocations. Another consideration to performance is working with assignment contours which are the timephased distribution of resource units on an assignment. There are two main categories of contours in Microsoft Project Professional: Predefined contours and edited contours. **Predefined Contours** —There are eight predefined contours. Each contour takes an assignment and divides it into ten segments. In each segment, work is calculated using the units predefined per that contour's definition. An **Edited Contour** is a contour that has been edited at the timephased level. Edited contours, however, differ in that they are not divided into ten segments of equal duration. First, the number of contour segments in an edited assignment contour depends on the actual edits made. The duration of each contour segment depends on the timescale in which the edit was made. Edited contours are a deviation from planned work (100% units), i.e. of not entering the exact hours against the set calendar. These contours have a large impact on the save time of projects to the server. Most customers experiencing performance related issues show a data profile with a high degree of statistical skew across projects relative to task and assignment custom field values. Typically a few projects have a high number of values and the remaining projects have very few. This data skew can impact the query plan logic. It may be required to create custom Plan Guides for your solution - http://technet.microsoft.com/en-us/library/ms190417.aspx Ensure there is a regular database maintenance plan running on SQL Server to provide faster insert and read operations. Fragmented indexes slow down the server performance retrieving the views for the users and queue operations in the back end. Refer to http://blogs.msdn.com/chrisfie/archive/2007/10/25/epm-database-maintenance-tasks.aspx for more details.
Calculation Factors	The table below represents key numbers to consider for scalar depth – the numbers presented are **not limits** – they **are thresholds** and represent the majority of the usage profile for Project Server that yields acceptable performance. There are many users who go beyond this threshold where additional optimization, functional design and process review are performed to maintain the desired throughput.

Object	Threshold
Number of assignments	**5 resource per task**
Number of assignment custom fields	**30**
Percentage of assignment schedule change	**20%**

† The appropriate architecture and physical design is required to support the thresholds noted in the table above.

When using Project Server with a different set of threshold values the impact can be observed throughout the solution from the client machine, system resources on the web front end, application server and SQL Server. For example in one of the performance testing labs we noted the web front end CPU required an additional 25% when increasing the number of assignments per task from 3 to 5.

Refer to the **PROJECT SYSTEM HEALTH REPORT V2.0** query in the appendix to monitor threshold values across the Project Server system.

Scope of Impact

- Client machine – additional RAM and CPU to manage larger project size and CPM calculations
- Active Cache – volatile schedules creating large data sets
- Web Front End – Additional CPU and RAM to manage node consistency scheduling across large data sets
- Application Server – Additional CPU and RAM to manage large data sets and queue operations
- SQL Server – Additional system resources and disk optimization to manage large data sets and queue operations
- The number of custom field values
- Views – Number of assignments per view and custom fields

D. Object: Resources

Definition

Enterprise resources are defined by their availability for assignments across multiple enterprise projects. There are several types of enterprise resources that can be defined: Work, Material, Cost, Budget, Generic, Team.

Guidelines for Acceptable Performance

Although there are published guidelines found at Plan for software boundaries (Project Server) http://technet.microsoft.com/en-us/library/cc197693.aspx these apply more to the client side boundaries. Cross dependencies with the server side

boundaries that need to be taken into consideration exist as well.

Only create resources in the system that will be performing project work. The more resources there are to manage the greater complexity there will be in the security association as well as scalar depth in the databases.

The total number of resources, as with the total number of projects, does not have a direct impact to performance of the system. It is the assignment of resources to tasks and projects where data complexity increases, and performance becomes impacted.

Calculation Factors

The table below represents key numbers to consider for scalar depth – the numbers presented are **not limits** – they **are thresholds** and represent the majority of the usage profile for Project Server that yields acceptable performance. There are many users who go beyond this threshold where additional optimization, functional design and process review are performed to maintain the desired throughput.

Object	Threshold
Number of resources	**40,000**
Number of resource custom fields	**30**

† The appropriate architecture and physical design is required to support the thresholds noted in the table above.

When using Project Server with a different set of threshold values the impact can be observed throughout the solution from the client machine, system resources on the web front end, the application server and SQL Server. For example in one of the performance testing labs we noted the web front end CPU required an additional 25% when increasing the number of assignments per task from 3 to 5.

Refer to the **PROJECT SYSTEM HEALTH REPORT V2.0** query in the appendix to monitor threshold values across the Project Server system.

Scope of Impact

- Server administrator managing frequent resource updates
- Security association across categories and groups
- Matrixed RBS association
- Security synchronization to Window SharePoint Services groups

E. Object: Custom Fields

Definition	Custom fields extend the attributes of tasks, resources and projects. Well defined enterprise custom fields can facilitate project object standardization in an organization. Enterprise custom fields are essentially unlimited – there is no hard-coded limit for number of enterprise custom fields, however, there are functional and performance thresholds that must be observed. Enterprise custom fields include the following data types: • Cost • Date • Duration • Flag • Number • Text
Guidelines for Acceptable Performance	Enterprise custom fields have a significant impact on performance of the system. As mentioned in the definition custom fields extend the core project objects and thereby make up the majority of the data in the object profile. Save times of projects, tasks and assignments, regardless of size, are consistent within the threshold parameters discussed thus far. It is the meta data (custom fields) and complex timephased contours that extend save times from seconds to several hours. The bulk of the custom field values also place contention on SQL in queue operations and potential table locking. The intended use of custom fields should be given great consideration when discussing and analyzing the organizations data requirements. How these fields will be surfaced in reporting, views and data warehousing are all vital areas to understand and discuss during the planning and design of the solution. The Task and Assignment objects referenced the statistical skew observed in many organizations where a minimal number of projects had a very large number of custom field values (many times over 100,000 each) with long project duration. The rest of the projects used minimal custom fields with small durations which affected the query plan for optimal data retrieval. It's recommended to distribute the custom field usage across all projects in the system to maintain a balanced data profile. For Analysis Services and Data Analysis views avoid selecting every custom field available for each cube type and reduce the number of dimensions created in the cubes. Create the calculated members with the PWA Cube configuration page and not within the Data Analysis Views. To improve queries against the Reporting Database consider creating indexes against columns that have a high use of custom fields for faster retrieval.

Organizations experiencing performance issues noted the use of "volatile" custom fields, namely the Flag data type and formulas which cause a recalculation of the values. For example, a custom formula with the Now() in the function will cause a recalculation of every custom field value it is associated with. This impacts the active cache, sets all tasks to be updated and bulk reload of the reporting database for that project, etc. Consider using the number data type with a 0/1 value in place of the flag data type. Limiting the use of volatile data types will decrease the quantity of data changed and improve throughput.

Limit the number of custom fields to 5 per view and 2 formulas per PWA view.

Calculation Factors

The table below represents key numbers to consider for scalar depth – the numbers presented are **not limits** – they **are thresholds** and represent the majority of the usage profile for Project Server that yields acceptable performance. There are many users who go beyond this threshold where additional optimization, functional design and process review is performed to maintain the desired throughput.

Object	Threshold
Active projects	**10,000**
Project duration	**24 months**
Number of inserted subprojects	**16**
Number of project custom fields	**30**
Total Enterprise custom field values	**32,000**
Number of baselines	**7**
Number of calendars	**5**
Number of tasks	**5,000 per project**
Task duration	**24 months**
Number of task custom fields	**30**
Number of assignments	**5 per task**
Number of assignment custom fields	**30**
Percentage of schedule change	**20%**
Number of resources	**40,000**
Number of resource custom fields	**30**
Percentage of schedule change	**20%**

† The appropriate architecture and physical design is required to support the thresholds noted in the table above.

When using Project Server with a different set of threshold values the impact can be observed throughout the solution from the client machine, system resources on the web front end, application server and SQL Server. For example in one of the performance testing labs we noted the web front end CPU required an

	additional 25% when increasing the number of assignments per task from 3 to 5. Refer to the **PROJECT SYSTEM HEALTH REPORT V2.0** query in the appendix to monitor threshold values across the Project Server system.
Scope of Impact	• Extended save times from Project Professional • Additional queue depth and table locking • TempDB operations • Disk IO contention • Timeouts in the Shared Service Provider, IIS and ASP.Net application • Long running queries • View timeouts in Project Web Access • Long running build time in OLAP Cube

F. Object: Security

Definition	The application security is defined in two areas: 1. Authentication - Managed by Windows SharePoint Services and provides user validation to the Content Root Site and Workspaces 2. Authorization - Managed by Project Server and provides granularity to ensure that a user has the necessary permissions required to perform their actions on Project Server data Permissions in Project Server is defined in two areas: 1. Global Permissions - Gives access to a feature within Project Server/Professional and applies across the server. It is not project or resource specific 2. Category Permissions - Gives access to data within objects through either project or resource permissions Security is defined in multiple areas within Project Server: 1. Users – at an individual level 2. Groups – users belonging to the group inherit the group permissions 3. Categories (data objects) - a category is a relationship between users/groups and data objects (Projects/Resources) 4. Views – data fields surfaced through the view and typically assigned to a category 5. RBS (Resource Breakdown Structure) - identifies the relationships between resources and associated projects with security rules set at each security category

Guidelines for Acceptable Performance	Authentication is not a performance issue; however, there is a synchronization process between the Windows SharePoint Services Root Site, Workspaces and Project Server users. During this synchronization process users are not able to log on to Project Server until their accounts are resynchronized to Windows SharePoint Services. Databases with a large number of users can take a considerable time to complete this action. A command line tool is available to set the sync process to meet the operational needs of the system. Typically changes to Global Permissions affect every user and changes to category permissions affect all users in that category. Do not make permission changes during business hours that may prevent user access to the system during the synchronization updates. The authorization process, also known as security trimming, takes place on the SQL Server tier and consumes a lot of TempDB operations. Mention has been made previously that you should ensure the appropriate TempDB file allocation and disk RAID configuration. Limit the number of additional categories that may be needed to meet your security requirements. Administrators as well as users high up on the RBS node who are authorized to see all the data in the system can expect longer operation times for views to load, and for updates to take place. Typically an administrators experience on the system is not indicative of Project Server performance. Applying the appropriate permissions for users at the category and view level, together with well defined views limiting custom field values improves calculation and rendering performance.
Calculation Factors	This is situationally dependant on the data profile and number of users. It's advisable to create more views with fewer columns than fewer views with a large column list. Limit custom fields to 5 per view and no more than 2 custom formulas per view.
Scope of Impact	• TempDB operations • Disk IO contention • Timeouts in the Shared Service Provider, IIS and ASP.Net application • Long running queries • Machine resources hosting Project Web Access handling large lists and views where insufficient CPU and memory are available.

3. Container: User

A. Object: Localization

Definition	There are 2 main types of users on the Project Server system. Project managers using Project Professional to manage the schedule, and team members using Project Web Access to report on their work assignments. User actions are tightly coupled with the Workload object, being the actions performed on the system.
Guidelines for Acceptable Performance	Determining performance is generally done by looking at the total number of users in the system; however, the table below under Calculation Factors illustrates typical time zone and regional distribution on the concurrency by user locale which effectively flattens peak load. A review of the user type, their locale and workload is important to designing a scalable system. Project managers using Project Professional should test out their physical location from the server to ensure Active Cache connectivity and save times meet acceptable throughput. There are situations with site-to-site connectivity where network performance is not optimal that may affect client performance and a Terminal Server environment is required. Referencing typical workloads of updating and republishing schedules within a narrow timeline may create excessive Queue depth and backend processing takes longer than usual. Spreading the update timeline will alleviate throttling the system with bulk updates. Team Members using Project Web Access should ensure their machines have the required system resources to process project data and render in acceptable timeframe. Also avoid pile-on situations of applying status/time updates right at the close of fiscal period.
Calculation Factors	Common requests to sizing a system come with the data point, given that they have (X) number of users on the system how many Web Front Ends do they need? This data point is irrelevant in sizing a system. All the factors need to be considered including concurrency model, data profile, scalar depth, application and SQL tier throughput. This table is a sampling of large real world large customers. The Regional Coverage column can be broken down into time zones which would further reduce user concurrency. To illustrate the sizing and throughput requirements we can compare **Customer C** with **Customer G**. **Customer C** has 3000 users located in the same region and therefore assume max concurrency rate of 3000. The scalar depth would be typical to what a 3000 user project team would create. **Customer G** has 15,000 users; however, the Max Concurrency model of EMEA with 45% is only 6,500 users. **Customer G** has 5 times the number of users than **Customer C** but only double the concurrency rate due to user distribution over alternate

regions. The scalar depth, however, may likely be 5 times that of **Customer C** and therefore, in this example, sizing guidelines should focus more on the backend processing throughput than front end tier handling user requests.

Customer	Users	Regional Coverage	Max Concurrency
A	6,000	EMEA: 50%; America: 30 %; Asia: 20%	3,000
B	3,000	EMEA: 50%; America: 27%; Asia: 23%	1,500
C	3,000	EMEA: 100%	3,000
D	3,000	EMEA: 80%; Asia: 20%	2,400
E	10,000	EMEA: 80% ; America: 20 %	8,000
F	10,000	EMEA: 50%; Asia 40%; America: 10%	5,000
G	15,000	EMEA: 43%; America: 27%	6,450
H	10,000	(Single Locale)	10,000
I	12,000	(Single Locale	12,000

Scope of Impact

- Peak load on system throttling the application server and web front end requests resulting with intermittent timeouts
- Delay in published data processing in the Queue
- Scalar depth unaccounted for in design of system
- Undersized user machines creating performance bottlenecks

Factor in load and concurrency when using external feeds to determine additional "extra load" in the back end.

B. Object: Workloads

Definition

Workloads defined in the context of this document are the articulated workflow of a specific user scenario.

Guidelines for Acceptable Performance

Different workloads affect the system at different times of the week based on the nature of the project management processes. It is important to identify these within the organization and alleviate pile-on situations placing peak load on the system in a very short amount of time. Examples of these are status updates from team members and schedule updates from project managers

Articulated workflow includes the process end to end cycle. For example: status

	updates that require approval, re-submitting, processing, and schedule updates and data synchronization to the Reporting Database. This entire workflow could take several hours due to wait times in the approval stages and needs to be considered when planning performance and throughput. Project managers should maintain schedules where updates typically only affect 20% of the data in the plan to reduce the cost of large saves and back end processing time. Consider the operational impacts of making changes during business hours. For example: changing global permissions will require all users re-sync to the root site and be temporarily unable to authenticate to the system. Doing this on a Friday afternoon during timesheet submission is not a good idea. Most organizations lean heavily on specific workloads to meet their business usage needs. Different workloads affect the system in different ways and it's important to review impacts to all the tiers in the solution when sizing and optimizing performance.
Calculation Factors	These should be reviewed based on the workload scenario. Typical patterns seen in the field across all organization with respect to peak activity are: • Friday: Team Member status update activity • Friday: Supporting Queue operations and Queue depth • Monday: Project manager schedule updates and publishing • Monday: Supporting Queue operations and Queue depth • Tuesday: Team Member review new assignments • Tuesday: Fiscal and status reports on system • Fiscal cutover: Adhoc burst activity The flow above illustrates Friday and Monday being the most active on the system.
Scope of Impact	• Queue depth and potential SQL locking • Disk I/O • Web Front Ends request contention in TempDB against Application tier usage • Data coherency in the articulated workflow – is the data processed through the system in time for the next scenario to begin • Interface feeds – additional load on the back end application tier unaccounted for in the system design • Lack of Operational maintenance to improve throughput (Index

	maintenance)

C. Object: Queue Job Processor Threads

Definition	The job processing threads reside within the "Queuing NT Service" process and the limit is set in the server administration area. Each thread spawned by job polling will retrieve and process waiting jobs in the queue.
Guidelines for Acceptable Performance	Queue Threads should be considered a type of “user” on the system. In fact, the total number of queue threads may, in many circumstances, place more load on the system than the entire user base coming through the web front end. Setting the number of threads (users) on the Queue affects many aspects of the design. As noted in the tables below under Calculation Factors there are diminishing returns when setting the thread count too high. This impacts SQL Disk I/O contention, TempDB processing, table lock escalation and Web Front End user request contention. There are five main areas to focus on when designing the system: • Maximum thread count to process queue depth during peak load in a given time window • RAID configuration to alleviate Disk IO contention on SQL Server • TempDB file allocation • Web Front End concurrency and security trimming against the Application Tier during peak load and queue depth • Regular Index Maintenance Plan to optimize CRUD operations General rules for maximum queue thread count is **not exceed 12** per instance. Monitoring the throughput, web front end requests and possible SQL contention within the system will identify the appropriate setting to use. Although it is possible to set a higher value, the Queue service will never spawn more than 20 threads total. Disable Fast Polling. Reduce the Polling Interval when the queue depth grows and potential table lock escalation is observed. Ensure the Application Server(s) have enough CPU and RAM to handle the queue operations. Use the performance counters to identify where the bottlenecks occur. It’s important to understand the data profile of the organization. The number of queue threads, project types, meta data values, etc has a significant impact to the individual processing time. One cannot expect a project of 500 tasks, 6 month duration and 10,000 custom field values to process in the same time of a project of 500 tasks, 5 year duration and 200,000 custom field values. Troubleshoot the organization’s performance issue using the **PROJECT SYSTEM HEALTH REPORT V2.0**

query sorting the results by Assignment CF values, Task CF values then Project Duration immediately identifies the top 10 projects raising the most complaints.

Calculation Factors

The following tables represent two real world examples of troubleshooting an organizations queue settings and trying to find the optimal count.

Example 1:

# Threads	Total Processing Time (minutes)	Projects/Sec	
8	18:15	3.65	N/A
16	13:35	3.32	N/A
20	23:30	4.7	N/A
24	23:50	4.77	N/A

Adjusting the thread count (factored by the number of CPUs at the Application Tier) illustrates diminishing returns. 16 threads provided the best throughput. There is little difference between 20 and 24 threads, as already stated, the Queue service will not spawn more than 20 threads regardless of the setting. Note this test occurred when there was no traffic on the Web Front End for user requests.

Example 2:

# Threads	Total Processing Time (minutes)	Projects/Sec	WFE Timeouts
4	>90 minutes	>36 sec/project	**2**
8	42:20 min	16.9 sec/project	**173**
12	32:05:00	12.8 sec/project	**651**

In this example, metrics were sampled with traffic on the Web Front End handling user requests. As the queue thread count was increased there were minimal gains in throughput due to an inefficient disk subsystem, however, the contention was noted with error 500 timeouts on the web servers. Throttling the queue needs to be balanced with servicing user requests on the front end for optimal throughput.

The thread count recommendation assumes having less than 10 instances in the Shared Service Provider.

Scope of Impact

- Disk IO subsystem
- Web Front End timeouts
- TempDB contention
- Database table lock escalation

	• Application Tier CPU and RAM • Index fragmentation

D. Object: Interface Feeds

Definition	Interface Feeds in Project Server are commonly integrated with external systems through the PSI (Project Server Interface) using web services. Interface Feeds should be considered a type of "user" on the system. In fact, they may, in many circumstances, place more load on the system than the entire user base coming through the web front end. Sizing a system should take these additional "user" into consideration.
Guidelines for Acceptable Performance	Determine the timing and frequency of the feeds to ensure a pile-on situation does not arrive during peak traffic on the system. Place logic in the feed to determine if the previous operation/update has completed before starting a new update specifically if the frequency of the feed is less than 60 minutes. Create a "pre-queue" repository outside of Project Server to batch data rather than create high traffic in the backend for frequent low volume operations. When submitting status through the PSI, the save time may be considerably longer than through the Project Professional client as the updates will use the Server Side Scheduling Engine to recalculate the schedule. Larger projects with a lot of extended meta data in the custom fields may take 5 times longer to process than a save directly from Project Professional which sends over data already rescheduled on the client scheduling engine. Ensure the timing of the feeds does not overlap with operational activities or place contention on other scheduled jobs. Size the Application Tier CPU and RAM to manage external integration load.
Calculation Factors	Dependant on the type of integration and data profile.
Scope of Impact	• Application Tier CPU and Memory • Queue operations and queue depth • Custom development adding "pre-queue" logic into the design • Data coherency – ensuring the data has processed through the entire articulated workflow before receiving the next update

E. Object: System

Definition	System requirements for users of Project Professional and Project Web Access.
Guidelines for Acceptable Performance	When troubleshooting performance issues in the field a common assumption by the system administrators is the issues result from the servers in the backend, however, perceived performance and user complaints are a result of under powered user machines **Project Professional** • Schedule data contains complex data structures that consume more memory than previous versions. Allow at least 1Gb of memory for Project Professional • Is a single threaded application. Multi-processors will not improve calculation time. Use the fastest processors available rather than adding more processors • Apply efficient project management processes and schedule design to limit the calculation overhead and large data transfers to the server which minimize the usage of the Active Cache **Project Web Access** • The Active X controls consume additional CPU to render the grid data. Multi-processors will not improve rendering time. Use the fastest processors available rather than adding more processors • Apply efficient security trimming through the security model of groups and categories • Limit the use of custom fields and calculated formulas in views (5 custom fields and 2 calculated formulas) • Create more views with fewer fields rather than single views with too many fields (more than 5 custom fields and 5 core fields) Administrators as well as users who are authorized to see all the data in the system can expect longer operation times for views to load, and updates to take place. Typically an administrators experience on the system is not indicative of Project Server performance experienced by other user roles. There is a performance cost when a user requests all the data for that operation. When a user complains about performance first isolate their local machine to ensure the system resources are sufficient, analyze the scalar depth of the data associated to that user and determine the security model/categories the user belongs to. Very often performance issues are resolved by optimizing the client machine without touching the server.
Calculation Factors	Project managers – 4Gb RAM, Dual CPU (fastest available)

	Project Web Access users – 2Gb RAM, Dual CPU (fastest available)
Scope of Impact	• Project Professional schedule calculations • Number of projects opened in a single sessions • View rendering time in Active X grids

4. Sizing Calculation Factors

The following section provides prescriptive guidance to sizing a Project Server 2007 solution. The guidelines assume a 3 tier architecture separating out the Web Front End, Application Server and SQL Server all running 64bit. In production a dedicated SQL Server is always recommended, however, combining the Web Front End and Application Servers is acceptable when ones considers the combined load for both servers roles and the consolidated server has enough system resources to respond to the application load.

There are several ways to calculate the size of the servers, and the steps below incorporate the major factors throughout the entire system together with the best practices presented within this document. The sizing process follows these sequential steps:

1. Determine the Scalar Depth of the system
2. Size the Application Server based on the Scalar Depth and desired throughput
3. Size the SQL Server against the Application Server and system resource ratios
4. Size the Web Front End against the User Concurrency
5. Determine the SQL file storage allocation

1. Sample Data Scalar Values

The **Project Ratio Mix** table provides a mix of project types in the majority of organizations sampled that were following standard project management processes. The **Estimated Database Size** table uses this ratio to calculate the Scalar Depth of a system based on the user profile.

Project Ratio Mix					
Type	**Ratio mix**	**Project duration**	**Number of Tasks**	**Task duration**	**Assignment to task ratio**
Small	25%	1m – 6m	100	1 – 60 days	3 – 5
Medium	50%	3m – 1yr	500	1 – 60 days	3 – 5
Large	20%	5m – 2yrs	1000	1 – 60 days	3 - 5

The **Estimated Database Size** table represents Scalar Depth and doubled in size for recovery operations. The database size includes all 4 databases, however, the sizes may vary within an organization where there may not be a requirement for Archive use and the project ratio mix varies. Planning should be given to the duration of the projects and active project retention policy within fiscal periods which will grow the size considerably over a period of time in production use. As noted in the Object: Database best practices a system in full production typically shows a size ratio off the Draft database of 1:2 for Archive, 1:2 for Published and 1:3 for Reporting. In other words, the Archive database is twice as big as Draft database, the Published database is twice as big as Draft database and the Reporting database is three times as big as the Draft database.

Estimated Database Size						
Profile	**Resources**	**Project Managers**	**Projects**	**Database Size**	**Recovery Disk Size**	**Total Disk Size**
One	1000	100	300	42Gb	42Gb	84Gb
Two	2500	250	750	63Gb	63Gb	126Gb
Three	5000	500	1500	86Gb	86Gb	172Gb
Four	7500	750	2250	129Gb	129Gb	258Gb
Five	10,000	1000	3000	193Gb	193Gb	386Gb
Six	15,000	1500	4500	289Gb	289Gb	578Gb

2. Application Server sizing

Size the Application Server against the Scalar Depth of the system, including any Interface Feeds, which will increase the "user load" on the Application tier. A simple formula below is used to calculate the desired job throughput within a 30 minute Queue processing time. The output is the CPU count which in turn allows the Memory and thread count to be calculated. The formula is as follows:

1. **Profile**: Use the Profile column the **Estimated Database Size** table to determine the number of projects in the system for processing.
2. **# Users**: Resource column from the chosen **Profile** row.
3. **# Jobs**: Project column from the chosen **Profile** row.
4. **JTM**: Calculate Jobs per Thread per Minute
    a. Any object being processed in the Queue is considered a job, e.g.: Project Save or Timesheet submit. Jobs execute on a single thread.
    b. In a variety of testing and field sampling the average job executes in **0.438 minutes**.
    c. Considering the job correlation of child jobs spawned from a parent operation, e.g.: A save job starts a publish job. A publish job starts an RDSSync job. Therefore the estimated time to complete a single correlation is **0.438 minutes / 3 = 0.146 minutes** (assuming multiple thread count in the Queue). **JTM = 0.146 sec**.
5. **HT (High Throughput):** To reduce a race condition on the Queue Group, set the Queue thread count to **1.5 per CPU** on the server (do not exceed more than 12 threads for the entire farm – the Queue thread count is per Application Server). For each Application Server role added to the farm, the Queue thread count must be reduced proportionately to ensure **Maximum Thread Count (MTC)** of 12 is not exceeded.
6. **PW (Processing Window):** The PW represents a processing window through the Queue during peak load of **30 minutes.**

7. **TPT (Total Processing Time):** The TPT represents the total estimated time to process all jobs on a single Queue thread - **JTM** x **# Jobs.**

8. **TTC (Total Thread Count):** The TTC represents the number of Queue threads to process the **# Jobs** within the desired processing window. **TPT / PW.**

9. **TCC (Total CPU Count):** The TCC represents the number of CPUs required for the Application Server. **HT / TTC.**

10. **TRC (Total RAM Count)**: The TRC represents the amount of memory for the Application Server. Apply a **1:2 ratio** of CPU to RAM.

Below is a sample scenario using **Profile Three** from the **Estimated Database Size** table. Applying the formulas above against this profile returns the CPU and RAM count together with the number of threads to set against the Queue.

Input	Value	Calculation
Profile	Three	N/A
# Users	5000	N/A
# Projects	1500	N/A
JTM	0.146	N/A
HT	1.5	N/A
TPT	219 minutes	JTM * #Projects
TTC	7.3 threads (Round up to 8 threads)	TPT / PW. TTC not to exceed 12.
TCC	4.66 (Round down to 4 CPU)	TTC / HT
TRC	8Gb	1:2

3. SQL Server sizing

Size the SQL Server against the hardware list from the Application Server calculated above. A simple formula below is used to determine the hardware list.

1. **TCC (Total CPU Count):** The TCC represents the number of CPUs required for the SQL Server. Apply a ratio of 1:2 against the Application Server **TTC**.
2. **TRC (Total RAM Count)**: The TRC represents the amount of memory for the SQL Server. Apply a **1:4 ratio** of CPU to RAM. This is imperative on a SQL Server using 64bit.
3. **Total TempDB Count:** The total number of TempDB files to allocate from the TCC. Apply a 1:1 ratio with the **TCC**.
4. **Total TempDB Size:** The total size of the TempDB to allocate from the Profile Database Size. The size is calculated as 25% of the Draft database size. As the Database Size column includes all 4 databases for that instance. **TempDB Size** = [**Profile Database Size**] X [**.25**] / [**TCC**]

5. **TempDB Spindle Count:** This represents the number of spindles to allocate across TempDB. The rule is one spindle per file group. RAID 10 is recommended, therefore doubling the count.

Following the sample scenario using **Profile Three** from the **Estimated Database Size** table as calculated for the Application Server the following formulas are applied to determine the SQL Server hardware list.

Input	Value	Calculation
Profile	Three	N/A
# Users	5000	N/A
# Projects	1500	N/A
Application Server TCC	4	Refer to Application Server table
SQL Server TCC	8	1:2
TRC	16Gb	1:4
Total TempDB count	8	1:1 with SQL Server TCC
Total TempDB size	2.6Gb	86Gb * 0.25 / 8
TempDB Spindle Count	16	8 * 2
SQL File Allocation - C:	80Gb	Operating System file. RAID 0 + 1.
SQL File Allocation - D:	120Gb	Local storage, log files. RAID 0 + 1.
SQL File Allocation - E:	110Gb	Data files RAID 10 or RAID 5: Database Size + 25% (Disk Defrag space).
SQL File Allocation - F:	30Gb	Log Files RAID 10: Total Database Size * 25%.
SQL File Allocation - G:	2.6Gb	TempDB Data files RAID 10.
SQL File Allocation - H:	1Gb	TempDB Log files RAID 10. Total TempDB Size * 25%.

4. Web Front End sizing

Size the Web Front End against the number of users as presented by the **Resources** column in the **Estimated Database Size** table. As demonstrated in the Microsoft Office Project Server 2007 Performance Testing Lab (http://www.microsoft.com/downloads/details.aspx?FamilyID=2f595e22-c9f1-4376-a8b2-7a7eccd2aba5&displaylang=en) whitepaper the greatest effect to the Web Front End in Project Server is the CPU spikes as more assignments are created against the relative tasks. Organizations using more than 5 assignments per task should monitor Web Front End system resources to determine if additional CPU is required by the Node Consistency and Task Scheduling component. A simple formula below is used to determine the hardware list.

1. **TCC (Total CPU Count):** The TCC represents the number of CPUs required for the Web Front End server.
    a. 1 – 1000 users = 2 CPU
    b. 1001 – 5000 users = 4 CPU
    c. 5001 – 15,000 users = 8 CPU

2. **TRC (Total RAM Count)**: The TRC represents the amount of memory for the Web Front End Server. Apply a **1:2 ratio** of CPU to RAM.

Following the sample scenario using **Profile Three** from the **Estimated Database Size** table the Web Front End Server the following formulas are applied to determine the hardware list.

Input	Value	Calculation
Profile	Three	N/A
# Users	5000	N/A
TCC	4	N/A
TRC	8Gb	1:2

Appendix

Available Project Server 2007 solutions

MSDN Code Gallery

MSND Code Gallery solutions	
Project Server 2007 Sample Databases	Project Server 2007 sample databases. The zip contains the following five databases: * Archive * Draft * Published * Reporting * Content
EPM 2007 Project Updater InfoPath Form	A web-enabled InfoPath 2007 Form using Project Server 2007 PSI .Net web services. The form allows project managers to easily mark tasks as complete in a sequential, process-like project plan…
Solution Connector for Microsoft Office Project Portfolio Server 2007	The Solution Connector is an ASP.NET web service and .NET class library which provides a set of methods for programmatically creating, updating or deleting projects and their attributes in …
Building an AJAX Web Part for Microsoft Office Project Server 2007	This sample contains a Project Server 2007 PSI Extension that retrieves geo data based on a custom field, and then the main feature is an AJAX Web Part that uses Virtual Earth to display the …
Microsoft Office Project Server 2007 PSI Extension Generator	PSI Extensions are custom web services that execute within the Project Server Interface infrastructure, sharing the same security context as other PSI web services. PSI Extensions can be very …
Project Server 2007 Migration Rename Tool	During a Project Server 2003 to Project Server 2007 migration, projects have "_Published" appended at the end of their name. This tool will enable you to "bulk" rename all projects and removed …
Project Server 2007 Lookup Table Update Control	Out of the box you cannot enforce control updates of specific lookup tables in Project Server 2007. This code sample leverages the standard Project Server interface API to limit who and what…
Microsoft Project Fx (mPfx) for Microsoft Project 2007	Microsoft® Office Project 2007 (Microsoft Project) underscores Microsoft's continued commitment to developers concerned with creating robust planning and scheduling tools for organizations with…
Project Server Interface 101 Development Samples	10 samples of how to do 101-level PSI development tasks
Project Server 2007 Performance Lab Kit	The Project Server 2007 Performance Lab Kit contains two Visual Studio solutions/tools to help you test your farm environment for capacity planning purposes: the "EPMTestTools" and the "EPM Stress …
Project Server 2007 Report Pack II - "The Top Reports"	The Project Server 2007 Report Pack II - published in August 2009 - provides reports that can be run in your EPM environment and enable easier access to EPM information that organization can …

CodePlex

CodePlex EPM solutions	
Project Server 2007 Timesheet & Statusing Customization Samples	Code samples that demonstrate how EPM (Project Server) 2007 Timesheet and Statusing functionalities could be customized using custom event handlers and Project Server Interface calls. These samples leverage the Timesheeting and Statusing API documented in the EPM 2007 SDK
Project Server 2007 VSTS Connector	Connector solution for Project Server 2007 and Team Foundation Server
Project Server 2007 Timesheet Tied-Mode Service and Event Handler	This project is used with Microsoft Project Server 2007. It consists of an NT service, an event handler, a test app and installer. The event handler is fired when a timesheet is saved. The handler saves the timesheet info in a SQL table. The NT service (TSAutoStatus) polls...
EPMSync Utility	Microsoft Office Project Server 2007 (Project Server) and Microsoft Office Project Portfolio Server 2007 (Portfolio Server) integrate via the Portfolio Server gateway. Data is exchanged by running either an import or an export from Portfolio Server. But this exchange is done...
Project Server 2007 Test Data Population Tool	The EPM (Project Server) 2007 Test Data Population Tool enables you to load large amounts of EPM data: resources, projects, tasks, assignments into a Project Server 2007 database. You can then use this data to test loads and help your organization plan for your Project Server 2
Project Server 2007 Queue Watch Tool	The Project Server 2007 Queue Watch Tool will help you monitor all queue activities for a specific Project Web Access (PWA) instance. This tool leverages the standard Project Server Interface publically documented web services to query and retrieve jobs in the Project Server ...
Project Server 2007 Log File Report Tool	The Project Server 2007 Log File Report Tool enables the import and the reporting of log file generated by your EPM & SharePoint farm. The Log File Report Tool will import log files (in an SQL database) from all servers in your farm with the ability to filter them by date. O...
Project Server and InfoPath 2007	This solution starter demonstrates how to leverage InfoPath and Forms Services as well Windows Workflow Foundation (WWF) hosted by SharePoint to create your own project initiation phase. The project initiation phase is different for every customers and organizations. It typic...

EPM Custom Fields Copy	The principal function of this project is to provide Project Server 2007 custom fields and lookuptables migration from development environment to production environment, using some PSI methods. I want to increase my initial project, adding more functionalities and solve some ...
Project Server 2007 Timesheet data population tool	The Project Server 2007 Timesheet Data Population Tool enables you to simulate timesheet entries in your farm. This tool can help you perform scalability studies of your PS architecture and validate the sizing of an existing architecture (by measuring timesheet queue throughput...
Search Project Server data using SharePoint Server's BDC and Enterprise Search	Search Project Server data using SharePoint Server's BDC and Enterprise Search
Project Reportcard	The Project Reportcard was developed as a tool to help Project Managers assess their projects based on organizational standards implemented by their PMO. Understanding what the organizational KPIs are going to look like before their project update is published can help a pro...
Project Server 2007 Auditing Solution Starter	EPM Auditing makes it easier to auditing and debug Project Server 2007 activities. You can audit multiple activities and output them to multiple sources. It's developed in C#.
Project Server Workspace Sync	This tool works with Project Server. It is a very simple tool that iterates over the list of Project Workspaces and triggers user sync for the members in the Windows SharePoint Services site and triggers the sync for issue, risks and deliverables with the reporting database.
Earned Value Add-In	The Earned Value Add-In has been developed as a tool to help project managers assess and visualize projects using earned value analysis.
Project Server 2007 AD/Resource Sync Utility	Demonstrates how to sync additional AD fields to resource custom fields for Project Server 2007.
Persisting SSAS OLAP Roles In Project Server 2007	During the standard Cube Building process in Project Server 2007, any manually added OLAP Roles in an OLAP database are deleted. The Cube building creates a default Role ProjectServerViewOlapDataRole that automatically adds all the Project Server users to this Role. This customize
Project Server 2007 Bulk Edit	The purpose of this project is to allow the bulk edit of resource data. With large numbers of resources Resource Center can take a long time to load, primarily due to the security checks required. This tool goes directly against the Project Server Reporting store to pull back t

Project Server 2007 Event Handler Admin Tool	This project contains code demonstrating how to use the PSI to add and remove event handler associations for Project Server 2007. Instead of displaying all possible event handlers like the PWA admin screens, this tool uses reflection to show only the implemented handlers and all
Project 2007 Test Framework	Controller/client test framework can be used to test any assembly or .exe with and object model (any MS Office app).
Project Server 2007 Timesheet AutoStatus Plus	Project Server 2007 Timesheet AutoStatus Plus is a major rewrite of Christophe Fiessinger's EPM tied-mode work. Planned enhancements are: multiple instance support, farm support, and elimination of the Windows Service in favor of a SharePoint Timer Job.

Data Capture

A. Performance Counters

The output of the capture should be in binary (**BLG**) format with a 24 hour capture period per file.

The Windows Server defines the performance data it collects in terms of objects, counters and instances. A performance object is any resource, application, or service that can be measured. Using Performance Monitor select performance objects, counters, and instances as listed in the tables below to collect and present data about the performance of the EPM system components. .

The counters are divided into three areas based on the EPM Server Role:

- Web Front End (WFE)—Runs the Windows SharePoint Services web site that hosts the Project Server farm; serves pages and content directly to users
- Application (APP)—Runs the Project Server Application Service and other application-tier services defined within Office Server
- Database (SQL)—Runs SQL Server and its related services

A machine that participates in an Office Server farm will participate in at least one of these roles. The matrix below describes the various server roles, and the counter sets that are appropriate for each. If a server participates in more than one role, simply collect the appropriate counters for *all* roles it performs.

Counter Set	WFE	APP	SQL
Generic	X	X	X
Project Server		X	X
SQL			X

Generic Performance Counters

Memory
Free System Page Table Entries
Pool Nonpaged Bytes
Pool Paged Bytes
Available MBytes
Pages/sec
Processor
Processor Time
Privileged Time
Interrupt Time
Network Interface
Bytes Total/Sec.
Output Queue Length
Packets/Sec.
Process
Total Time (pick processes appropriate to server role)
Private Bytes
Thread Count
Virtual Bytes
Physical Disk
% Disk Time
Average Disk Queue Length
Current Disk Queue Length
Disk Bytes/Sec.
Disk Reads/Sec.
Disk Writes/Sec.
System
Processor Queue Length
Context Switches/sec

Project Server Performance Counters

ProjectServer:QueueGeneral

% Sql Retries / Day
% Sql Retries / Hour
Active Job Processing Threads
Average Unprocessed Jobs / Day
Current Unprocessed Jobs
New Jobs / Minute
Sql Calls / Hour/Day
Sql Calls / Minute
Sql Retries / Minute
ProjectServer:Winproj
Average time taken for Project Open
Percentage of incremental save to full save
Winproj full open count in the last hour
Winproj full save count in the last hour
Winproj incremental open count in the last hour
ProjectServer:QueueJobs

% Jobs Failed / Day
% Jobs Failed / Hour
% Jobs Retried / Day
% Jobs Retried / Hour
Average Processing Time / Day
Average Processing Time / Minute
Average Wait Time / Day
Average Wait Time / Minute
Jobs Failed / Minute
Jobs Processed / Hour/Day
Jobs Processed / Minute
Jobs Retried / Minute
ProjectServer:User Activity
PSI Calls per Second

SQL Server Performance Counters

Access Method
Page Splits/Sec
Buffer Manager
Buffer Cache Hit Ratio Total Pages
General Statistics
User Connections
Locks
Average Wait Time (ms) Target Server Memory (KB) Total Server Memory (KB)
Memory Manager
Target Server Memory (KB) Total Server Memory (KB)
SQL Statistics

Batch Requests/Sec. SQL Compilations/Sec.

B. Unified Logging Service (ULS) - Logs

All products based on SharePoint technologies provide a built-in logging engine named Unified Logging System (ULS). It allows the applications and related component (Microsoft or third-parties) to log activity to the Windows application Event Log and/or to a log file on each server running SharePoint.

Log Location

The log files are, by default, located under C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\LOGS, the file names always start with a prefix consisting in the name of the server they were generated on: <servername>-<output-format>.log.

Depending upon their configuration, some event may also be logged in the Windows Application event log.

Log File Format

The log files expose the following fields:

1. Timestamp: Equivalent to the “TimeGenerated” field in the “Application” event Log
2. Process: the image name of the process logging its activity followed by its process ID (PID) between parentheses. Interestingly, IIS worker processes may also log their activity, they are therefore logged under w3wp.exe
3. TID
4. Area: This maps the “Source” field in the “Application” event Log
5. Category: this maps the “Category” field in the “Application” event Log
6. EventID: A unique internal Event ID
7. Level
8. Message
9. Correlation: may contain a link to the EventID of another logged event

C. Internet Information Services (IIS) - Logs

The output of the capture should be the W3C Extended Log File format. The W3C Extended log file format is the default log file format for IIS. It is a customizable ASCII text-based format. You can use IIS Manager to select which fields to include in the log file, which allows you to keep log files as small as possible.

Internet Information Services (IIS) offers a number of ways to record the activity of your Web sites. IIS logging is designed to include information such as who has visited your site, what was viewed, and when the information was last viewed.

Log Location

The log files are, by default, located under %SYSTEMROOT%\System32\LogFiles\W3SVC**.log

Log File Format

Before you collect data you must enable *all* fields in this table and extended properties that are not being logged on the Web servers in your farm. You should enable it on all Web servers in the farm.

The log files expose the following fields:

Field	Appears as
Time taken	time-taken
Host	cs-host
Date	Date
Time	Time
Client IP address	c-ip
Method	cs-method
URI stem	cs-uri-stem
URI query	cs-uri-query
HTTP status	sc-status
User agent	cs(User-Agent)
User name	cs-username
Server IP address	s-ip
Server port	s-port

Protocol substatus	sc-substatus
Service Name and Instance Number	s-sitename
Server name	s-computername
Win32 status	sc-win32-status
Bytes sent	sc-bytes
Bytes received	cs-bytes
Protocol version	cs-version
Cookie	cs(Cookie)
Referrer	cs(Referrer)

D. Windows Server Event Log

A copy of the event logs should be made in the standard event (**EVTX** or **EVT**) format from each server in the farm. An example of an output file name is Servername – log type – date.EVTX

From the Event Viewer, you are able to monitor and create a copy of the Application and System events that were recorded by the Windows applications and Windows operating system, respectively.

Log Location

The log files are, by default, located under C:\Windows\system32\config\EVENTTYPE.Evt or %SystemRoot%\System32\Winevt\Logs\EVENTTYPE.Evtx .

Log File Format

All event levels, Critical, Warning, Verbose, Error and Information from any source should be copied from the Application and System logs.

PROJECT SYSTEM HEALTH REPORT V2.0

Copy and paste this query into SQL Server query analyzer to run a report on system health and determine scalar depth across the data profile.

```
/*

-- ********************************************** SUMMARY NOTES
********************************************** --

PROJECT SYSTEM HEALTH REPORT V2.0
```

Extracts a series of data points from Draft & Reporting databases to qualify the data in a Project system.

Run the report against your Draft database and save the results to file, send the resulting .CSV file with

Column Headers to Microsoft (We are working on making the data analysis public as a series of reports to

remove the need to send data to Microsoft)

CUSTOMIZATION

1. Set local variable @DRAFT_DB_NAME with your DRAFT database name.

1. Set local variable @PUBLISHED_DB_NAME with your PUBLISHED database name.

3. Set local variable @REPORTING_DB_NAME with your REPORTING database name (and SERVER if on a separate server).

4. Set local variable @ERADICATE_PROJ_NAME to 1 if require masking project name for privacy information

PRIVACY/INFORMATION PROTECTION NOTES

1. Report contains data that can be used to quantify the amount of project work in the org,

please review this for sensitivity prior to sending the data to Microsoft

CHANGELOG

01/05/2009 pmc: Prepared for Customer Release, some p

01/05/2009 pmc: Changes made for Project Server 14 Alpha (Project Server 2010)

05/11/2009 EPMGP: Added values to incorporate Quality Assurance data points for the Health Check Tool

*/

-- ********** BEGIN DECLARE VARIABLES **********

```
DECLARE @SQL_SELECT nvarchar(max);          -- Variable to contain SELECT statements
DECLARE @SQL_JOINS nvarchar(max);           -- Variable to contain JOIN statements
```

```
DECLARE @PS_TASK_START varchar(20);                -- Variable to swith column name between PS2007 and
PS2010 implementation

DECLARE @PS_TASK_FINISH varchar(20);               -- Variable to swith column name between PS2007 and
PS2010 implementation

DECLARE @DRAFT_DB_NAME varchar(100);               -- Variable to set the DRAFT database name

DECLARE @PUBLISHED_DB_NAME varchar(100);           -- Variable to set the PUBLISHED database name

DECLARE @REPORTING_DB_NAME varchar(100);           -- Variable to set the REPORTING database name

DECLARE @ERADICATE_PROJ_NAME CHAR(1);              -- Variable to switch between displaying project name or
masking the value


---------------------------------------------------  END DECLARE VARIABLES  ---------------------------------------------------


-- ********************************************** BEGIN SET VARIABLE VALUES
********************************************** --


SET @DRAFT_DB_NAME = 'BLANK_Draft';                -- Set local variable @DRAFT_DB_NAME with your DRAFT
database name

SET @PUBLISHED_DB_NAME = 'BLANK_Published';        -- Set local variable @PUBLISHED_DB_NAME with your
PUBLISHED database name

SET @REPORTING_DB_NAME = 'BLANK_Reporting';        -- Set local variable @REPORTING_DB_NAME with your
REPORTING database name

SET @ERADICATE_PROJ_NAME = 0;                      -- Set local variable to 1 to mask project name for privacy
or default to 0

SET @PS_TASK_START = 'TASK_START_DATE';            -- Set local variable to TASK_START_DATE for PS2007 and
REM PS2010

SET @PS_TASK_FINISH = 'TASK_FINISH_DATE';    -- Set local variable to TASK_FINISH_DATE for PS2007 and REM
PS2010

--SET @PS_TASK_START = 'TASK_SCHED_START';         -- Set local variable to TASK_SCHED_START for PS2010
and REM PS2007

--SET @PS_TASK_FINISH = 'TASK_SCHED_FINISH';       -- Set local variable to TASK_SCHED_FINISH for PS2010
and REM PS2007
```

```
---------------------------------------------------- END SET VARIABLE VALUES ----------------------------------------------------

SET @SQL_SELECT = N'use [' + @DRAFT_DB_NAME + N']

SET NOCOUNT ON;

WITH ProjHierarchy
AS (SELECT Proj.PARENT_PROJ_UID, 1 AS DEPTH
   FROM dbo.MSP_PROJ_HIERARCHIES AS Proj
   UNION ALL
   SELECT Proj.PARENT_PROJ_UID, Hier.DEPTH+1
   FROM dbo.MSP_PROJ_HIERARCHIES AS Proj
   INNER JOIN ProjHierarchy AS Hier
   ON Proj.CHILD_PROJ_UID = Hier.PARENT_PROJ_UID)
SELECT
                    Proj.PROJ_UID AS [PRJ: PROJ ID]
    ,CASE ' + @ERADICATE_PROJ_NAME + N' WHEN 0 THEN Proj.PROJ_NAME ELSE N''*** NAME ERADICATED ***''END AS [PRJ: NAME]
    ,CASE(Proj.PROJ_TYPE)
                        WHEN 0 THEN N''Project''
                        WHEN 1 THEN N''Template''
                        WHEN 2 THEN N''Global''
                        WHEN 3 THEN N''Resource Global''
                        WHEN 4 THEN N''LightWeightProject''
                        WHEN 5 THEN N''Inserted Project''
```

```
                    WHEN 6 THEN N''Master Project''
                    WHEN 100 THEN N''New Project''
                    WHEN 101 THEN N''New Template''
                    WHEN 102 THEN N''New Global''
                    WHEN 103 THEN N''New Resource Global''
                    WHEN 1000 THEN N''Inactive Project''
                    WHEN 1001 THEN N''Inactive Template''
                    WHEN 1002 THEN N''Inactive Global''
                    WHEN -1 THEN N''Void''
                    ELSE ''Unknown''
            END AS [PRJ: TYPE]
,ISNULL(CollabRes.RES_NAME, Proj.PROJ_PROP_AUTHOR) AS [PRJ: AUTHOR]
,ISNULL(CONVERT(VARCHAR(10), Proj.CREATED_DATE,111), ''0000/00/00'') AS [PRJ: CREATED]
,ISNULL(CONVERT(VARCHAR(10), Proj.PROJ_INFO_START_DATE,111), ''0000/00/00'') AS [PRJ: START_DATE]
,ISNULL(CONVERT(VARCHAR(10), Proj.PROJ_INFO_FINISH_DATE,111), ''0000/00/00'') AS [PRJ: FINISH_DATE]
,DATEDIFF(mm, Proj.PROJ_INFO_START_DATE,Proj.PROJ_INFO_FINISH_DATE) AS [PRJ: DURATION (Mnth)]
,ISNULL(CONVERT(VARCHAR(10), Proj.MOD_DATE,111), ''0000/00/00'') AS [Project Modified]
,ISNULL(CONVERT(VARCHAR(10), Proj.PROJ_INFO_STATUS_DATE,111), ''0000/00/00'') AS [PRJ: STATUSED]
,ISNULL(Task.TASK_PCT_COMP, 0) AS [PRJ: %COMPLETE]
,CASE
                    ISNULL(Proj.PROJ_SESSION_UID,N''00000000-0000-0000-0000-000000000000'')
                    WHEN N''00000000-0000-0000-0000-000000000000''
                    THEN 0 ELSE DATEDIFF(d,Proj.PROJ_CHECKOUTDATE,getdate())
            END AS [PRJ: CHECKED_OUT(Days)]
,CAST(Task.TASK_ACT_WORK / 60000 AS DECIMAL(10,2)) AS [PRJ: TOTAL_ACTUAL_WORK(Hrs)]
,CAST(Task.TASK_REM_WORK / 60000 AS DECIMAL(10,2)) AS [PRJ: TOTAL_REMAINING_WORK(Hrs)]
```

```
,ISNULL(LinkData.[Cross Project Links],0) AS [PRJ: #_CROSS_PROJECT_LINKS]

,ISNULL(HierSummaryData.[Number of SubProjects],0) AS [PRJ: #_SUB_PROJECTS]

,ISNULL(HierSummaryData.[SubProject Depth],0) AS [PRJ: SUB_PROJECT_DEPTH]

,ISNULL(AllTaskData.[Task Count],0) AS [TSK: TOTAL_TASK_COUNT]

,ISNULL(RecentTasks.[Recent Changes],0) AS [TSK: TOTAL_RECENT_CHANGE]

,ISNULL(CONVERT(DECIMAL(18,0), CONVERT(FLOAT, RecentTasks.[Recent Changes])/CONVERT(float,
AllTaskData.[Task Count])*100), 0) AS [TSK: %_CHANGED]

,ISNULL(SRA.[SRA Count], 0) AS [TSK: SRA_COUNT]

,ISNULL(LeafTaskData.[Leaf Task Count],0) AS [TSK: LEAF_COUNT]

,ISNULL(LeafTaskData.[Max Leaf Calendar Duration (Days)], 0) AS [TSK:
MAX_LEAF_TASK_CALENDAR_DURATION(Days)]

,ISNULL(LeafTaskData.[Average Leaf Calendar Duration (Days)], 0) AS [TSK:
AVERAGE_LEAF_TASK_CALENDAR_DURATION(Days)]

,ISNULL(LeafTaskData.[Total Leaf Task Ignoring ResCalendar], 0) AS [TSK:
TOTAL_LEAF_TASK_IGNORING_RES_CALENDAR]

,ISNULL(LinkData.[Total Number of Links],0) AS [TSK: TOTAL_NUMBER_OF_LINKS]

,ISNULL(AssnDataAggregated.[Average Resources on Assigned Tasks],0) AS [ASN:
AVG_RESOURCES_ON_ASSIGNMENTS]

,ISNULL(AssnDataAggregated.[Total Number of Assignments],0) AS [ASN: #_TOTAL_ASSIGNMENTS]

,ISNULL(AssnDataAggregated.[Number of Tasks with Assignments],0) AS [ASN:
TOTAL_TASK_WITH_ASSIGNMENTS]

,ISNULL(RealData.[Actual Count],0) AS [ASN: TOTAL_ASNBYDAY_COUNT]

,ISNULL(ProjCF.[Number of Project Custom Field Values],0) AS [CF: PRJ_CF_VALUES]

,ISNULL(ProjLCF.[Number of Local Custom Field Definitions],0) AS [CF: PRJ_LOCAL_CF_DEFINITIONS]

,ISNULL(TaskCFAggregated.[Total Task Custom Fields],0) AS [CF: TOTAL_TASK_CFs]

,ISNULL(TaskCFAggregated.[Max Task Custom Fields],0) AS [CF: MAX_TASK_CFs]

,ISNULL(TaskCFAggregated.[Average Task Custom Fields],0) AS [CF: AVG_TASK_CFs]

,ISNULL(TaskCFAggregated.[Number of Tasks with Custom Fields],0) AS [CF: #_TASK_WITH_ CFs]
```

```
    ,ISNULL(AssnCFAggregated.[Total Assn Custom Fields],0) AS [CF: TOTAL_ASN_CFs]

    ,ISNULL(AssnCFAggregated.[Max Assn Custom Fields],0) AS [CF: MAX_ASN_CFs]

    ,ISNULL(AssnCFAggregated.[Average Assn Custom Fields],0) AS [CF: AVG_ASN_CFs]

    ,ISNULL(AssnCFAggregated.[Number of Assn with Custom Fields],0) AS [ASN: #_ASN_WITH_CFs]

    ,ISNULL(TaskBaseAggregated.[Number of Baselines],0) AS [BSL: #_BASELINES]

    ,ISNULL(TaskBaseAggregated.[Total Task Baseline Rows],0) AS [BSL: TOTAL_TSK_BASELINE]

    ,ISNULL(AssnBaseAggregated.[Total Assn Baseline Rows],0) AS [BSL: TOTAL_ASN_BASELINE]

    ,ISNULL(ResBaseAggregated.[Total Resource Baseline Rows],0) AS [BSL: TOTAL_RES_BASELINE]

    ,ISNULL(Resources.[Total Resources],0) AS [RES: TOTAL_TEAM]

    ,ISNULL(Resources.[Enterprise Resources],0) AS [RES: TOTAL_ENT_TEAM]

    ,ISNULL(ActiveResAggregated.[Active Resources],0) AS [RES: TOTAL_ACTIVE]

    ,ISNULL(CONVERT(VARCHAR(10), Proj.WPROJ_LAST_PUB,111), ''0000/00/00'') AS [PWS: PUBLISHED]

    ,ISNULL(Collab.WPROJ_STS_SUBWEB_NAME, 0) AS [PWS: NAME]

    ,ISNULL(Collab.PROJ_TOTAL_DOC_COUNT, 0) AS [PWS: DOC_COUNT]

    ,ISNULL(Collab.PROJ_ACTIVE_ISSUE_COUNT, 0) AS [PWS: ISSUE_COUNT]

    ,ISNULL(Collab.PROJ_ACTIVE_RISK_COUNT, 0) AS [PWS: RISK_COUNT]

    ,ISNULL(CONVERT(DECIMAL(18,2),(CAST(AssnDataAggregated.[Total Number of Assignments] AS REAL) /
AllTaskData.[Task Count])),0) AS [RATIO: ASN:TSK]

    ,ISNULL(CONVERT(DECIMAL(18,2),(CAST(AssnCFAggregated.[Total Assn Custom Fields] AS REAL) /
AssnDataAggregated.[Total Number of Assignments])),0) AS [RATIO: ASNCF:ASN]

    ,ISNULL(CONVERT(DECIMAL(18,2),(CAST(TaskCFAggregated.[Total Task Custom Fields] AS REAL) /
AllTaskData.[Task Count])),0) AS [RATIO: TSKCF:TSK]

    ,CONVERT(VARCHAR(10), GetDate(),111) AS [RECORDED_DATE]'


SET @SQL_JOINS = N'


        FROM dbo.MSP_PROJECTS AS Proj WITH(NOLOCK)
```

```
-- ************  Project Summary Task
INNER JOIN dbo.MSP_TASKS AS Task WITH(NOLOCK)
ON (Proj.PROJ_UID = Task.PROJ_UID AND Task.TASK_OPTINDX = 1) -- Use the Project Summary Task for rollup values

-- ************  Count the number of resource assignments on summary tasks
LEFT OUTER JOIN (Select MP.PROJ_UID AS PROJ_UID, COUNT(DISTINCT MA.Task_name) AS [SRA Count]
                 from msp_assignments as MA inner join msp_projects as MP on MA.Proj_uid=MP.proj_uid
                 where MA.task_uid in (select task_uid from msp_tasks where task_is_summary=1)
                 GROUP BY MP.PROJ_UID) AS SRA
ON (PROJ.PROJ_UID = SRA.PROJ_UID)

-- ************  Get the Project Manager Name
LEFT OUTER JOIN ' + @PUBLISHED_DB_NAME + N'.dbo.MSP_RESOURCES As CollabRes
ON Proj.PROJ_PROP_AUTHOR = CollabRes.WRES_ACCOUNT

-- ************  Get the rowcount from our (usually) largest table
LEFT OUTER JOIN (SELECT Assn.ProjectUID as PROJ_UID,COUNT(*) + 1 AS [Actual Count]
                 FROM ' + @REPORTING_DB_NAME + N'.dbo.MSP_EpmAssignmentByDay AS Assn
                 GROUP BY Assn.ProjectUID) AS RealData
ON (Proj.PROJ_UID = RealData.PROJ_UID)

-- ************  Project Workspace data from PublishedDB
```

```
LEFT OUTER JOIN ' + @PUBLISHED_DB_NAME + N'.dbo.MSP_PROJECTS AS Collab
ON      (Proj.PROJ_UID = Collab.PROJ_UID)

-- ************  Project Custom Field Data
LEFT OUTER JOIN (SELECT pcf.PROJ_UID,COUNT(*) AS [Number of Project Custom Field Values]
                                    FROM dbo.MSP_PROJ_CUSTOM_FIELD_VALUES AS pcf WITH(NOLOCK)
                                    GROUP BY pcf.PROJ_UID) AS ProjCF
ON (Proj.PROJ_UID = ProjCF.PROJ_UID)

-- ************  Project Local Custom Fields
LEFT OUTER JOIN (SELECT plcf.PROJ_UID
                                        ,COUNT(*) AS [Number of Local Custom Field Definitions]
                                    FROM dbo.MSP_PROJECT_CUSTOM_FIELDS AS plcf WITH(NOLOCK)
                                    GROUP BY plcf.PROJ_UID) AS ProjLCF
ON (Proj.PROJ_UID = ProjLCF.PROJ_UID)

-- ************  Project Hierarchy Below each Project
LEFT OUTER JOIN (SELECT HierData.PROJ_UID
                                        ,COUNT(HierData.DEPTH) AS [Number of SubProjects]
                                        ,MAX(HierData.DEPTH) AS [SubProject Depth]
                                    FROM (SELECT Proj.PROJ_UID
                                                    ,Hier.DEPTH
                                        FROM ProjHierarchy as Hier
                                        INNER JOIN dbo.MSP_PROJECTS AS Proj WITH(NOLOCK)
                                        ON Hier.PARENT_PROJ_UID = Proj.PROJ_UID) AS HierData
                                    GROUP BY HierData.PROJ_UID) AS HierSummaryData
```

```
        ON (Proj.PROJ_UID = HierSummaryData.PROJ_UID)

        -- ************  Project resource data
        LEFT OUTER JOIN (SELECT Pres.PROJ_UID
                                        ,COUNT(*) AS [Total Resources]
                                        ,SUM(CAST(Pres.RES_IS_ENTERPRISE_RESOURCE AS INT)) AS
[Enterprise Resources]
                                        ,SUM(CAST(Pres.RES_BOOKING_TYPE AS INT)/2) AS [Proposed
Resources]
                                FROM dbo.MSP_PROJECT_RESOURCES AS Pres WITH(NOLOCK)
                                WHERE Pres.RES_ID > 0 -- Ignore internal resources
                                GROUP BY Pres.PROJ_UID) AS Resources
        ON (Proj.PROJ_UID = Resources.PROJ_UID)

        -- ************  Resources with Assignments
        LEFT OUTER JOIN (SELECT ActiveRes.PROJ_UID
                                        ,COUNT(*) AS [Active Resources]
                                FROM (SELECT Assn.PROJ_UID
                                                ,Assn.RES_UID
                                        FROM dbo.MSP_ASSIGNMENTS AS Assn WITH(NOLOCK)
                                        INNER JOIN dbo.MSP_PROJECT_RESOURCES AS Pres
WITH(NOLOCK)
                                        ON (Assn.PROJ_UID = Pres.PROJ_UID AND Assn.RES_UID
= Pres.RES_UID)
                                        WHERE Pres.RES_ID > 0 -- Ignore internal resources
                                        GROUP BY Assn.PROJ_UID, Assn.RES_UID) AS ActiveRes
                                GROUP BY ActiveRes.PROJ_UID) AS ActiveResAggregated
```

```
 ON (Proj.PROJ_UID = ActiveResAggregated.PROJ_UID)


-- ************  Leaf task data
LEFT OUTER JOIN (SELECT Task.PROJ_UID
                                    ,COUNT(*) AS [Leaf Task Count]
                                    -- IMPLEMENTATION NOTE - Use _SCHED_ below for PS2010, Ignores User Scheduled dates
                                    ,MAX(DATEDIFF(d,Task.' + @PS_TASK_START + N', Task.' + @PS_TASK_FINISH + N')) AS [Max Leaf Calendar Duration (Days)]
                                    ,AVG(DATEDIFF(d,Task.' + @PS_TASK_START + N', Task.' + @PS_TASK_FINISH + N')) AS [Average Leaf Calendar Duration (Days)]
                                    ,SUM(CAST(TASK_IGNORES_RES_CAL AS INT)) AS [Total Leaf Task Ignoring ResCalendar]
                                FROM dbo.MSP_TASKS AS Task WITH(NOLOCK)
                                WHERE Task.TASK_IS_SUMMARY = 0x0 -- Ignore summary tasks
                                AND Task.TASK_IS_MILESTONE = 0x0 -- Ignore milestones
                                AND Task.TASK_IS_SUBPROJ = 0x0 -- Ignore subprojects
                                GROUP BY Task.PROJ_UID) AS LeafTaskData
ON (Proj.PROJ_UID = LeafTaskData.PROJ_UID)


-- ************  All visible tasks in the project
LEFT OUTER JOIN (SELECT Task.PROJ_UID
                                    ,COUNT(*) AS [Task Count]
                                FROM dbo.MSP_TASKS AS Task WITH(NOLOCK)
                                WHERE Task.TASK_OPTINDX > 1
                                GROUP BY Task.PROJ_UID) AS AllTaskData
ON (Proj.PROJ_UID = AllTaskData.PROJ_UID)
```

```
-- ************  Task data changed on Last Project saved
LEFT OUTER JOIN (SELECT Task.PROJ_UID
                                ,COUNT(*) AS [Recent Changes]
                        FROM dbo.MSP_TASKS AS Task WITH(NOLOCK)
                        INNER JOIN dbo.MSP_PROJECTS AS Proj WITH(NOLOCK)
                        ON (Task.PROJ_UID = Proj.PROJ_UID AND Task.MOD_REV_COUNTER =
Proj.MOD_REV_COUNTER)
                        WHERE Task.TASK_OPTINDX > 1
                        GROUP BY Task.PROJ_UID) AS RecentTasks
ON (Proj.PROJ_UID = RecentTasks.PROJ_UID)

-- ************  Task Custom Field Data
LEFT OUTER JOIN (SELECT TaskCF.PROJ_UID
                                ,MAX(TaskCF.[Task CF Count]) AS [Max Task Custom Fields]
                                ,AVG(TaskCF.[Task CF Count]) AS [Average Task Custom Fields]
                                ,SUM(TaskCF.[Task CF Count]) AS [Total Task Custom Fields]
                                ,COUNT(*) AS [Number of Tasks with Custom Fields]
                        FROM (SELECT TCF.PROJ_UID
                                        ,TCF.TASK_UID
                                        ,COUNT(*) AS [Task CF Count]
                                FROM dbo.MSP_TASK_CUSTOM_FIELD_VALUES AS TCF
WITH(NOLOCK)
                                GROUP BY TCF.PROJ_UID, TCF.TASK_UID) AS TaskCF
                        GROUP BY TaskCF.PROJ_UID) AS TaskCFAggregated
ON (Proj.PROJ_UID = TaskCFAggregated.PROJ_UID)
```

```
-- ************  Link Data
LEFT OUTER JOIN (SELECT Links.PROJ_UID
                                        ,COUNT(*) AS [Total Number of Links]
                                        ,SUM(CAST(LINK_IS_CROSS_PROJ AS INT)) AS [Cross Project
Links]
                                FROM dbo.MSP_LINKS AS Links
                                GROUP BY Links.PROJ_UID) AS LinkData
ON (Proj.PROJ_UID = LinkData.PROJ_UID)

-- ************  Task Baseline Data
LEFT OUTER JOIN (SELECT TaskBaseData.PROJ_UID
                                        ,COUNT(*) AS [Number of Baselines]
                                        ,SUM(TaskbaseData.[Task Baseline Rows]) AS [Total Task
Baseline Rows]
                                FROM (SELECT TaskBase.PROJ_UID
                                                ,TaskBase.TB_BASE_NUM AS [Baseline]
                                                ,COUNT(*) AS [Task Baseline Rows]
                                        FROM dbo.MSP_TASK_BASELINES AS TaskBase WITH(NOLOCK)
                                        GROUP BY TaskBase.PROJ_UID, TaskBase.TB_BASE_NUM) AS
TaskBaseData
                                GROUP BY TaskBaseData.PROJ_UID) AS TaskBaseAggregated
ON (Proj.PROJ_UID = TaskBaseAggregated.PROJ_UID)

-- ************  Assignment Baseline Data
LEFT OUTER JOIN (SELECT AssnBaseData.PROJ_UID
                                        ,COUNT(*) AS [Total Assn Baseline Rows]
```

```
                                        FROM dbo.MSP_ASSIGNMENT_BASELINES AS AssnBaseData
WITH(NOLOCK)
                                        GROUP BY AssnBaseData.PROJ_UID) AS AssnBaseAggregated
        ON (Proj.PROJ_UID = AssnBaseAggregated.PROJ_UID)

        -- ************  Resource Baseline Data
        LEFT OUTER JOIN (SELECT ResBaseData.PROJ_UID
                                        ,COUNT(*) AS [Total Resource Baseline Rows]
                                        FROM dbo.MSP_PROJECT_RESOURCE_BASELINES AS ResBaseData
WITH(NOLOCK)
                                        GROUP BY ResBaseData.PROJ_UID) AS ResBaseAggregated
        ON (Proj.PROJ_UID = ResBaseAggregated.PROJ_UID)

        -- ************  Assignment Data Summary
        LEFT OUTER JOIN (SELECT AssnTaskData.PROJ_UID
                                        ,AVG(AssnTaskData.[Assignment Count]) AS [Average Resources
on Assigned Tasks]
                                        ,SUM(AssnTaskData.[Assignment Count]) AS [Total Number of
Assignments]
                                        ,COUNT(*) AS [Number of Tasks with Assignments]
                                FROM (SELECT Assn.PROJ_UID
                                                ,Assn.TASK_UID
                                                ,COUNT(*) AS [Assignment Count]
                                        FROM dbo.MSP_ASSIGNMENTS AS Assn WITH(NOLOCK)
                                        GROUP BY Assn.PROJ_UID, Assn.TASK_UID) AS AssnTaskData
                                GROUP BY AssnTaskData.PROJ_UID) AS AssnDataAggregated
         ON (Proj.PROJ_UID = AssnDataAggregated.PROJ_UID)
```

```
	-- ************  Assignment Custom Field Data
	LEFT OUTER JOIN (SELECT AssnCF.PROJ_UID
				,MAX(AssnCF.[Assn CF Count]) AS [Max Assn Custom Fields]
				,AVG(AssnCF.[Assn CF Count]) AS [Average Assn Custom Fields]
				,SUM(AssnCF.[Assn CF Count]) AS [Total Assn Custom Fields]
				,COUNT(*) AS [Number of Assn with Custom Fields]
			FROM (SELECT ACF.PROJ_UID
					,ACF.ASSN_UID
					,COUNT(*) AS [Assn CF Count]
				FROM dbo.MSP_ASSN_CUSTOM_FIELD_VALUES AS ACF
WITH(NOLOCK)
				GROUP BY ACF.PROJ_UID, ACF.ASSN_UID) AS AssnCF
			GROUP BY AssnCF.PROJ_UID) AS AssnCFAggregated
	ON (Proj.PROJ_UID = AssnCFAggregated.PROJ_UID)
	ORDER BY
		[CF: TOTAL_ASN_CFs]
		,[CF: TOTAL_TASK_CFs]
		,[PRJ: DURATION (Mnth)]
'

EXEC (@SQL_SELECT + @SQL_JOINS)
```