Estimating Capacity Requirements

A critical step in the planning of a Yellowfin deployment is estimating the capacity requirements. This task approximates the initial computing resources needed to meet delivery objectives and service level agreements. Capacity estimation can be achieved through either an informal or a structured approach and should include consideration of the key activities undertaken by users.

The main system requirements that should be estimated are:

- **Processing power:** Processing power is generally the most critical resource as it is the most expensive. This involves determining the number of physical computer servers, the speed, bit size, and number of CPUs.
- **System memory:** The memory requirements of the planned implementation can influence the choice of the operating system and, in turn, the processing power decision. For example, any application process running on a 32-bit Windows platform can use only ~3 GB of system memory. In contrast, 64-bit Windows, UNIX, and Linux operating systems support a theoretical 18 exabyte address space for application use.
- **Hard disk space:** In terms of computing resources, hard disk resources represent the cheapest of the three types of system resources and are also the least critical.

Capacity estimation should be an ongoing exercise to ensure that the system infrastructure continues to support and grow with users' needs. Managing user requests, retrieving data from data sources, adding analytical functionality, formatting and distributing report results, and performing report interactions all use computing resources. An increase in the number of users, the number of reports, the volume of data, and the amount of analytical reporting handled by the Yellowfin servers also increase the amount of computing resources needed to maintain similar levels of performance and report throughput. Capacity estimation gives a reasonable idea of the amount of computing resources needed to power a Yellowfin implementation.

Capacity Factors & Their Influence

The most important factors in capacity planning include:

- Number of users
- Design of reports
- Interactivity of reports
- Monitoring and administration
- Data Preparation
- Using Stories
- Using Signals

Their impact on the three main system requirements can be summarized as follows:

Capacity Estimation Factors	Processor	Memory	Disk Space
Number of Users			
Total Users			✓
Active Users		✓	
Concurrent Users	✓	✓	
Data Preparation			
Transformation Design	✓	✓	✓
Transformation Execution	✓	✓	
Reports Design			
Report Size		✓	✓
Analytical Complexity	✓	✓	
Report Layout	✓	✓	
Report Interactions			
Reporting Browsing & Creation	✓	✓	

Batch Reporting & Delivery	✓	✓	✓
Report History			✓
Implementation and Administration			
Metadata		✓	✓
Caching		✓	✓
Monitoring	✓	✓	✓
Stories			
Story Design		✓	✓
Story Interaction	✓	✓	
Signals			
Signal Execution	✓	✓	✓
Signal Interaction	✓	✓	

Estimating the Number of Users

The total number of users of an application is the easiest factor to estimate. Since user information, audit trails and unique report instance information is stored in the Yellowfin repository, increasing the number of users increases the hard disk requirements. Depending on how they use the application, users also have a secondary impact on the CPU and memory requirements.

At any given time, only 10-20% of all users concurrently use a typical reporting application. The Yellowfin application server allocates memory and address space for each active user from a shared pool of system resources. The larger the number of concurrent user sessions, the more memory resources consumed by the application server.

Typically, only 30-50% of the users logged into Yellowfin run jobs simultaneously. Thus, Yellowfin has a user activity ratio of 3-10% of the total named users. Active users directly influence the processing and memory resources of the computer servers.

As a rule of thumb, allow a minimum **8 GB of memory and 3 GHz (8 Thread) processor per 25 concurrent users** on the application server.

As an example, if there were 500 named users in a Yellowfin instance, this would suggest that 50 active users are logged in at any given time (with significantly more for peak usage). This would imply the need for a minimum of a 16 thread server with 16 GB of memory. Of course, these are only guidelines and resource consumption will vary depending upon several factors – this includes the size and complexity of the database and reports being run, your online vs. background workload characteristics, etc.

Each installation is different. It is highly recommended that performance tests be performed with a workload approximating the target production environment. This will ensure that the software meets the performance requirements and Yellowfin content loads in a reasonable time.

Understanding Report Characteristics

The number of reports deployed is an important factor in determining capacity requirements. However, the number of reports alone does not provide an accurate estimation; the size, analytical complexity, and report layouts must also be considered.

Report Size

Recently run reports are stored in cache using memory resources. The amount of memory used depends on the amount of data and data types used, and this can be estimated relatively accurately. In addition, reports where history is kept (i.e. an XML copy of the result set is stored in the Yellowfin repository each time the report is refreshed) will also utilize hard disk resources.

Analytical Complexity

Yellowfin adds analytical capabilities to reports by generating sectioned and cross-tab data, adding calculations and subtotals, drilling, sorting, pivoting, and managing filter prompts. Reports that require a large amount of analytical processing require more processing power and memory, but the exact amount can be difficult to quantify.

Report Layout

Yellowfin formats reports in many ways such as cross-tab grids, graphs, HTML, PDF, and CSV. Applying formats to report results uses both processing power and memory. The more formatting properties, controls, and report objects contained on a report, the more processing and memory resources are consumed.

Incorporating the Effects of Report Interactions

Users will have varying levels of interaction with Yellowfin from receiving reports via email to complex reporting functions. The effect of these interactions can be categorized as follows:

Report Browsing and Creation

Yellowfin provides a very rich reporting, analysis, and monitoring environment that is centrally maintained. The platform also contains a broad range of functionality; from basic activities such as accessing the system, finding, running, and printing reports, to powerful analytical features such as drilling, sorting, pivoting, adding calculations, and exporting. Sophisticated report development capabilities are also available in this platform.

The quantity and variety of report interactions consume both processing power and memory on all tiers on the web infrastructure – web servers, application servers, Yellowfin repository / database server, the data sources, and the web browser. Capacity should be estimated for all the tiers.

Batch Reporting and Delivery

Many organizations process commonly used reports and scheduled reports in batches. Ideally, this occurs during off-peak hours and periods of low system usage. The key constraints that govern system resources for batch reporting are the quantity of reports executed and delivered, and the size of the batch window in which the processing occurs. Shorter batch windows and more reports require more processing and memory resources on the application server to complete report processing and delivery in the allotted time.

Report History

Report result sets can be stored in the Yellowfin repository for reuse by multiple users – this provides the ability to compare multiple versions of the report. This also reduces the load on the data sources but will increase the hard disk requirements of your Yellowfin installation.

Cleanup processes and deletion policies should be set by your administrators to reduce the total resources needed.

Administering and Monitoring BI Implementations

Yellowfin provides rich application metadata with numerous monitoring and performance enhancing features to ensure successful implementations. Increasing functional richness and continued performance monitoring take up system resources that must be included during capacity estimation.

Metadata

The size of the central Yellowfin repository has an influence on the memory consumption of the application server. Upon startup, the application server loads metadata configuration information into memory.

Caching

Yellowfin provides comprehensive caching capability to improve throughput, optimize query performance, and reduce end user response times. Server caches are stored in memory.

Monitoring

Yellowfin logs statistics and user activity in the Yellowfin repository / database server. Storing the usage and performance statistics consumes a negligible amount of processor and memory resources, but additional hard disk space is required to store the record.

In addition, log files are also created by Yellowfin. A capacity estimation exercise must account for performance monitoring and usage statistics logging. It is important to note that these monitoring resources should be used to improve future capacity estimation and tuning activities.

Yellowfin Signals

The automated analysis performed by Yellowfin Signals can be as complex or as simple, depending on a number of factors. The Signal analysis factors that influence your system, are discussed below.

Signals Execution

Depending on the size of each dataset to be analyzed, Signals analysis will require more or less memory in order to analyze an entire time slice at once. This means that the number of rows returned by each slice would affect the amount of memory used. The number of unique dimension values in the selected dimension fields combined with the size of the analysis date scope are the greatest contributing factors to the maximum memory usage of each analysis sub-task.

Signals is optimized to be highly parallel, meaning that it can take advantage of multiple cluster nodes and CPU cores in order to break down the problem into smaller chunks to be analyzed separately. For this reason, usually you will hit database, network, or memory constraints before CPU speed/power becomes the bottleneck for Signals, however this can become a constraint on very low-power environments, or environments which analyze a lot of Signals tasks or subtasks concurrently.

The last concern for Signals execution is disk space. Signals can affect configuration database size greatly, as quite a lot of metadata about the Signals and their setup is stored, as well as the large number of notifications which can be generated on a system with many Signals users and/or jobs. Caching datasets also affects this, but it is configurable and optional.

Signals Interaction

Depending on some factors such as dataset size, number of related and correlated lines, etc., interacting with a Signal can be relatively taxing on the system as a whole. A Signal is essentially an automatically generated Dashboard in the sense that it runs several reports with charts whenever you open it. This means that memory and CPU usage can spike when many users are concurrently looking at Signals.

Signals dataset caching can help slightly, but generating charts and loading dataset(s) into memory still needs to happen, and caching mostly just helps with loading times if you have a slow source. This stage can often be more taxing than running the Signals analysis itself, since more datasets need to be loaded simultaneously.

The Assisted Insights analysis which is presented with each Signal is something which only runs once when the first user opens the Signal. Each time after that, the cached and pre-rendered version is loaded and no report is run. This is to avoid running and storing many extra Assisted Insights reports for Signals which may never be opened. This can, however, also affect the system with the same concerns that a normal Assisted Insights analysis would have.

Previous topic: Server requirements
Next topic: Server configuration