Enterprise Job Scheduling Evaluation Guide and Checklist

How to Choose an Enterprise-Class Job Scheduling Solution

How to Choose an Enterprise-Class Job Scheduling Solution

Purpose

Among the handful of disciplines routinely practiced in the data center, job scheduling may be the most important of all. This is a bold statement, given that job scheduling competes with other important systems management functions like backup and recovery, network management, and systems security. While these are important disciplines in their own right, there is little dispute that, depending on the size of the enterprise, a job scheduler is routinely managing thousands - and in many cases, tens of thousands - of individual mission-critical business processes every day. In fact, the number of processes involved can be so large that a manual approach is not possible.

Custom job scheduling solutions that rely on native operating system utilities such as CRON, NT Scheduler, PERL and VB scripts quickly collapse under the weight of their own unmanageable “spaghetti code” and custom scripting requirements. Given this backdrop, it is easy to see how automated job scheduling solutions are an indispensable part of the data center.

This document discusses the requirements for job scheduling software in complex enterprise environments that feature multiple, interconnected enterprise-class applications that touch every aspect of the organization. It provides a comprehensive list and description of vital job scheduling features that are critical for the smooth and efficient delivery of business processes. A companion checklist is also available to assist anyone actively evaluating job scheduling and event-driven workload automation software. The checklist is a useful tool for quickly and efficiently gathering information on products under consideration.

To set the proper context for a detailed discussion of key job scheduling functions, this document includes a brief history of the job scheduling marketplace. This background will help the reader understand how changing business and industry requirements impact the job scheduling environment.

Background

The discipline of job scheduling was first established in the 1970s when it became a key strategic infrastructure component for large mainframe-oriented data centers. A variety of products were created and extensively marketed, and it became widely accepted that a reliable job scheduling tool was required to effectively manage batch-oriented business processes on the mainframe.

During this period of mainframe-centric computing, a common understanding of the key features of job scheduling began to emerge. When UNIX began to make inroads into the data centers in the mid-1990s, IT managers widened their search for job scheduling solutions to manage batch processing in these new distributed computing environments.

As the shift to UNIX-based applications began, only a few mainframe job scheduling vendors expanded their offerings to meet the demands of this new environment. Instead, mainframe vendors and many mainframe-oriented data centers experimented with managing distributed workload from the mainframe. As UNIX servers continued to make their way into the core data center, a new group of competitors entered the market with products created expressly for managing distributed job scheduling.

As the two competing approaches were deployed, it quickly became apparent that the products created expressly for the distributed environment were a better approach, and the mainframe approach to managing distributed workload was relegated to a small market segment. Even so, many mainframe data centers still cling to the elusive dream of managing all job workloads, regardless of platform, from a single mainframe console. Unfortunately, few if any have been able to achieve this goal and few vendors appear to focus on the issue.

Evolution of the Distributed Job Scheduler

The early solutions for distributed computing environments proved to be reasonably robust and did a passable job of mimicking mainframe scheduling features and functions, but they suffered from being first to the marketplace. Eventually, all these first-generation products failed in terms of scalability, flexibility, and ease-of-use, and were ultimately acquired in the late 1990s by mainframe companies still grappling with the unique issues and challenges of managing jobs in the ever-increasing distributed computing environments.

During this consolidation phase of the job scheduling marketplace, new competitors began to emerge who were focused on developing more advanced solutions for job scheduling in the distributed environment. These newcomers had several distinct advantages over their legacy counterparts.

Improved Development Technologies

The early distributed scheduling products were generally challenged because of the limitations of the base technology used to develop them. By the early to mid-90s, development tools for the distributed environment had not matured sufficiently to be of significant value, so products from that period were generally developed with older, harder-to-maintain code platforms and used the UNIX Motif standard for their graphical user interface. In fact, a number of the first-generation products were actually ported from other proprietary platforms in the early 1990s, with the usual complications caused by porting software products. The combination of the older, inflexible development environments, lack of robust software design tools, and lack of any GUI standards created problems as the market began to mature and the requirements for distributed scheduling began to increase. These first-generation products could not keep pace with changing requirements and began to falter in terms of scalability, ease-of-use, ease-of-installation, configurability, and maintainability.

An Understanding of Cross-Platform Development

Like their mainframe predecessors, the early UNIX products essentially had a single-platform orientation, the only difference being that the platform was UNIX instead of MVS or OS/390. This problem came into focus when Microsoft Windows 95 and 98 took over the desktop and Windows NT and 2000 servers became more widely used in the data center. While the early UNIX batch scheduling products had the initial lead on Windows NT and 2000, their shortcomings for true cross-platform, distributed scheduling became evident as more and more companies attempted to incorporate Windows NT or 2000 into their data center operations.

The latest generations of job scheduling products are capable of exploiting the ease of use of the Windows desktop and recent software development technologies. These products also reflect a deeper understanding of the rigorous architectural and systemic requirements for building a true cross-platform solution that incorporates UNIX, Windows, and other platforms.

Scalability

The latest generation of scheduling products has been developed in an era that has large distributed application workloads already running in the enterprise. Early schedulers had yet not been confronted by the size and volume of workloads that we see commonly today. When the requirements for increased scalability surfaced, the early scheduling tools did not have the power and design to manage large, distributed workloads.

These shortcomings did not become readily apparent for several years, at which point their users were forced to look for replacements. Many data centers are in this position today and are looking to replace their first-generation job schedulers because of scalability and maintenance issues.

The latest generation of scheduling solutions is perfectly positioned to meet the rigorous demands of the modern, distributed data center operations. Unfortunately for the first- and second-generation solutions, it is much easier to design a product to be scalable from inception than it is to try to retrofit scalability into an older design.

An Understanding of true 24/7/365 Operations

When mainframe job schedulers were first created, there were two distinct periods of operation in the data center - online and batch. Typically during the day, databases were “online” and only available to end users for entering transactions at their terminals. During off-peak hours (typically at night), the databases were brought “offline” and batch processing activity (typically reporting and other batch-intensive jobs) was allowed to commence. This batch-processing window was often as long as 12 hours (for example, from 6 p.m. to 6 a.m.), but sometime after the first generation of distributed scheduling products was introduced into the market, business demands began to shrink this “batch window.”

By the late 1990s, when the latest generation of job scheduling products were being created, two significant issues had started to change the landscape for corporations and software providers alike: namely, globalization and e-business. This shift in the business landscape began to overwhelm data center operations by demanding true 24/7/365 operations.

The net effect was the elimination of the batch window and the creation a new set of mission-critical requirements for batch scheduling tools. The modern job scheduling solution is able to respond to these evolving requirements with the addition of new features for improved scalability and faster workload management. Job schedulers now need to respond to a wide variety of events for launching and managing jobs rather than merely relying on the traditional date and time scheduling model. A number of these business-critical features, including event management, are discussed later in this document.

Requirements of Modern Job Scheduling and Workload Automation Solutions

Over the years, the job scheduling function has matured as a discipline and has become accepted as a requirement for any comprehensive infrastructure management strategy. During this maturation phase, the industry has developed an understanding of what core functions are mandatory for a product to be considered an enterprise-grade job scheduling solution. This section of the evaluation guide discusses the core set of features commonly acknowledged as requirements for any enterprise-class workload automation tool.

At a strategic level, any scheduling solution must provide a means to configure targets and meet deadlines that are aligned with business objectives and their attached service- level agreements (SLAs). The primary reason to employ an automated scheduling solution is to deliver SLAs faster and to create efficiencies that allow IT resources to be deployed toward projects that help the business innovate. The first question that needs to be asked is: “Has the solution been proven in real-world situations to automate the alignment of IT needs with the needs of the business?”

Scalable and Flexible, Fault-Tolerant Architecture

N-Tier architectures are flexible enterprise-class service delivery systems to consider when evaluating a scheduling solution that can deliver on the business needs for performance, extensibility, and scalability. With its separated logical layers, an N-Tier architecture is a flexible framework that allows the scheduling product to handle heavy workloads while maintaining client stability and reliability. It can also support a flexible deployment model so that users in remote regions can have a dedicated scheduling access that is closer to their region, which therefore yield faster response times.

In this model, the central processing engine (CPE) is focused primarily on interacting with the scheduling database and driving the workloads. In this highly flexible environment, input and interaction is done through a browser and handled by a middle-tier web server, which in turn manages the flow of information to and from the CPE. This multilayered and highly flexible architecture separates core functions and helps keep processes running smoothly while supporting the actions of high numbers of distributed and concurrent users.

When considering architecture, key requirements to assess are flexibility and fault tolerance. Flexibility is a measure of how easy the product is to deploy in a given environment and how quickly new environments can be added or changed. Can the CPE be installed on any server? Can agents and/or application adapters be deployed on most machines in the computing environment? Is there an agentless option? Can more than one master scheduling node be running, enabling the various masters to interact in such a way as to provide increased flexibility or fault tolerance?

Some vendors have attempted to disperse scheduling data processors throughout the data center - making each scheduling node the equivalent of the CPE. However, the issues associated with error recovery in the event of machine or communications failure have largely ruled out this type of architecture as a serious choice. Because the vast majority of jobs are dependent on other job outputs on other machines, there is little justification for moving scheduling data around the network when it can be maintained more reliably and securely on a central resource.

Support for a Wide Range of Modern Enterprise Applications

The primary reason for implementing a job scheduling solution is to accurately and efficiently deliver mission-critical applications processing. This requirement existed for homogeneous computing environments and still exists today.

What has complicated modern SLA delivery is the requirement to successfully integrate with many of today's enterprise applications - SAP, PeopleSoft, Oracle EBS, JD Edwards, Informatica, Cognos, and so on. Unfortunately, the feature set and functionality of most scheduling tools is insufficient to manage job steps originating from other vendor-supplied and custom developed applications within a large enterprise. The ability to connect to a wide variety of distributed business application solutions allows the enterprise to:

- Run any end-to-end business workflow, including downstream business intelligence (BI) reporting and analysis tools.
- Rapidly incorporate other application steps into their data center business processes requirements so that enterprisewide logic can be automated.
- Perform discovery of the options available for the environment, so that configuration options are automatically tailored with the correct options. This eases configuration and allows for faster configurations with fewer errors.

In addition, to minimize overhead and service concerns, these job scheduling vendors should ideally provide an option to run processes using agentless technology - having no footprint on the target machine.

Some job scheduling vendors have extended their products to try to encompass the needs of these critical applications and have created interfaces that make calls directly to the core scheduling functionality. There are commercially available interfaces that integrate these adapter solutions, and it might be worthwhile to evaluate them. At the same time, not all adapter interfaces are created equal, so take the time to understand how a vendor actually develops and certifies their interface:

- Is the interface current with the latest application APIs?
- What specific features are available to support the application?
- Does the solution use vendor approved integration points?
- Can native service-oriented architecture (SOA) jobs be invoked using REST Web Services methods, Simple Object Access Protocol/Web Service Definition Language (SOAP/WSDL), or both?

In addition to managing jobs on traditional enterprise resource planning (ERP) platforms, enterprise schedulers must take into consideration connections to applications developed in Java and those developed for SOA platforms. These custom interfaces should support a breadth of Java Message Service (JMS), Java Management Extensions (JMX), and Web Service connectivity, so scheduling operators can take advantage of these messaging and job scheduling capabilities to incorporate Java-based applications, SOAP-, and REST-based Web Service methods into the broader scheduling environment.

A viable scheduling solution should also integrate seamlessly with the latest business intelligence (BI), data warehouse (DW), extract, transform, load (ETL), and data backup and recovery applications. These applications process data that span the entire enterprise and that come from multiple applications, to deliver the business insight required to make the right strategic decisions at the right time. Managing dependency logic and automating job step sequencing is critical for the delivery of relevant and timely information for strategic decision making. To achieve these goals, four unique states are necessary:

- Broad enterprisewide visibility of the entire application, storage, and data warehouse landscape
- The ability to automate a wide range of functions from dependency mapping, to application and data warehouse job steps, to event and alerts management
- Support for BI application specific user interfaces that saves re-keying of jobs already defined
- Granular historical and real-time visibility and analytics into the entire job stream critical path

By integrating an efficient and error-free workload automation solution into the companywide decision-making environment, IT operations staff can define connections and dependencies, scheduling the correct business process in the right order to help ensure that the correct mission-critical data is always delivered to the right place at the right time.

Database Support

Another aspect of enterprisewide workload coverage is database support. On the mainframe, it was and is commonplace to use “flat” or indexed files as the underlying data repository for all scheduling objects and rules. When these products were developed and in widespread use, relational databases were not widely used; in fact, DB2, now the leading mainframe database product, did not arrive until the late 1980s. Early distributed scheduling tools followed the flat-file model because they lacked reliable relational database technology. However, when relational technology arrived in the middle 1990s, the leading products adopted it.

Another benefit of using relational database technology is the ability to mirror databases, if desired, and incorporate mirroring into fault tolerance, backup and error recovery strategies, further facilitating unattended operations.

FTP, SFTP, and FTPS Support

Increasingly, businesses are working to integrate their internal operations and to develop closer links with partners and customers. Companies in financial services, insurance, healthcare, e-commerce, and retail depend on seamless data exchange to integrate their operations with partners or service providers. The technology most commonly used to exchange data or transfer files among applications and partners is the File Transfer Protocol (FTP).

Because there is a close relationship between data exchange and an organization's batch processes, both must be seamlessly and simultaneously managed to attain the high levels of automation, security, speed, recovery, and accuracy required. However, support for FTP is not enough, because FTP is inherently insecure and transfers data without encryption. User names, passwords, and FTP commands can be viewed with packet sniffers. Consequently, a viable scheduling solution needs to support both FTP and SSH File Transfer Protocol (SFTP) or FTP over SSL (FTPS).

Modern workload automation solutions should not require scripting to support automatic file transfers as part of the batch business processes. These solutions should natively recognize when files arrive via FTP or SFTP/FTPS and trigger one or more events to move and process the data.

Comprehensive Programming Interfaces (APIs)

In traditional job scheduling environments, the scheduler was the controlling entity and executed or invoked other applications. In today's complex and distributed computing environments, it is often desirable for other applications to interface with, or "call," the scheduler directly in order for the scheduler to provide services on the application's behalf. Therefore, a full-featured, unified API for programmatic integration is necessary in an enterprise solution in order for IT staff to bring all necessary applications, systems, and technologies into this single interactive SLA delivery model. Enterprise-class schedulers provide comprehensive programming interfaces that easily facilitate integration with other applications. These can be invoked through command-line, API libraries callable from languages such as VB, C++, and Java messaging services.

As part of the API, there should be a command-line interface (CLI) for UNIX and Windows environments allowing scheduling interactions from a remote system. A CLI-based scheduling approach is important because it allows an alternative means to create and change jobs and can be used to automate job definitions and updates, which can be faster than using the standard user interface. Jobs can also be "programmed" to take advantage of the CLI execution syntax, giving users the ability to dynamically insert or update future jobs. Web Service interfaces are also becoming increasingly important to expose scheduling functions as a set of services in a SOA environment.

Today's heterogeneous data centers include composite service-oriented architecture (SOA-based) applications. This means that business process management (BPM) engines expose workflows as web services that are callable through standard REST-based methods and SOAP/WSDL protocol support. To function reliably, workflows in these environments must integrate flawlessly with job streams defined as part of a larger jobs- processing environment. The scheduler should simplify workloads in these environments and provide visibility and control over jobs of all magnitudes and complexities across all operations.

Platform Coverage

Because a typical distributed enterprise includes more than one type of hardware and operating system, it is a requirement to thoroughly analyze current and future needs with regard to the scheduler's platform and technology support. A vendor should support the data center's current and future operating system requirements and have a history of staying current when operating system updates occur. It is also important to consider the data center's historic and projected preferences for computing technology. For instance, although Windows is the clear choice for the desktop, can the job scheduling CPE reside on UNIX or Windows server? Ideally, being positioned to support both UNIX and Windows is the best strategy, because needs may change over time.

Also consider whether popular application operating environments are supported including Windows, UNIX, Linux, OpenVMS, OS/400, z/OS, HP NonStop, and others. A z/OS data center will also likely want the flexibility of an enterprise solution that provides enterprisewide integration with their mainframe scheduling solution.

When OS-level executable agents can't be deployed to manage systemwide workflows because of security issues, machine ownership, or resource constraints, do the solutions being evaluated support agentless scheduling as well? Agentless approaches allow scheduling to be extended to environments where installation of an agent may not be allowed due to company policy or due to constrained system resources.

As an example, can Windows applications be managed using a remote job adapter function, or look should you look agentless SSH capabilities for scheduling on Linux, UNIX, and other networked devices? With an agentless solution, network connectivity is only used when a job is delivered to the target. Also, users do not have to wait for a certified agent to be installed to schedule on these resources. What's more, system administrators do not need to deal with installing or deploying agents if the necessary resources are going to be used only to run a small handful of jobs. When considering platform coverage also consider whether scheduling in virtualized environments can be achieved through the support for standard virtualization technologies.

Graphical Management

In computing environments, it is a requirement that scheduling products provide true graphical management of the scheduling environment. This can take a variety of forms, but should include an easy-to-understand central console from which all jobs and schedules can be viewed. This list of jobs should be indexed and filterable so that only certain types of jobs are visible or can be sorted to the top of the list. Additionally, the console should be color-coded so that certain jobs - failures, in particular - can be highlighted for easy viewing. As the console gets more sophisticated, it might have other indicators to tell users at a glance if one or more jobs are in an error state or completed normally.

More sophisticated products have dashboard features that present the ongoing results of scheduled activity in graphical formats such as line, pie, Gantt, and PERT charts. These charts can give novice and experienced users alike a quick view of the progress of the daily schedule, the number of jobs that have finished successfully/unsuccessfully, the percentage of the daily schedule that has already executed, and other common measurements.

In addition to an easy-to-use consoles, scheduling solutions should present job streams graphically to make it easier to understand job status and interdependencies. These pictorial representations make the creation and troubleshooting of complex workflows significantly easier to manage. In addition, a modern scheduler should provide graphical approaches to defining jobs and job streams, job dependencies, alerts, events, and other scheduling tasks, doing away with the need for command-line interactions or the creation of custom scripts.

In today's computing environment, it is standard practice for applications to have a browser-based user interface, and job schedulers are no exception. A web-enabled UI for a job scheduling tool simplifies administration of the tool itself and also gives operations personnel the flexibility to log in anytime, anywhere, to monitor and control their scheduling environment from any location. When looking at a product's web UI capabilities, it is also important to know just how much of the product is available through the web browser, because some tools have only limited capabilities in this area and force users to use a fixed, fat-client UI for an expanded set of features.

Business Calendars and Time Zones

At the heart of any job scheduler is the concept of starting jobs at a particular date and time. Companies rely on many types of calendars to drive their business: fiscal calendars, manufacturing calendars, payroll calendars, and holiday calendars that drive specific aspects of modern business operations.

Enterprise job schedulers are expected to provide comprehensive calendaring functionality that is easy to use, provides highly visible forecasting data, and permits calendar combinations that achieve date-driven business processing requirements. For example, the calendaring functionality should accommodate rules for what day payroll should run if it happens to fall on a holiday (with different locations having different holiday schedules). Not so surprisingly, complex calendars are required to manage a modern corporation, with some companies managing several hundred, depending on their size and number of geographic locations.

Enterprise business processes are often global in nature, with various steps of the process running in different parts of world that operate on various time zones. Cross-time-zone scheduling is a factor as modern businesses operate in multiple theaters and must be aware of time zone idiosyncrasies, like the fact that daylight savings time changes on irregular intervals. Cisco® Tidal Enterprise Scheduler can account for these situations and give the user the ability to specify the time zone at a per-job basis. Doing so will help ensure that the job will launch correctly no matter where the job resides and run as expected.

Because of the vital nature of these features, solution buyers are advised to scrutinize the calendar and time zone capabilities to see that they meet their needs in terms of flexibility, ease of use, maintenance, and complexity.

Dependencies

Following calendaring functionality, in terms of importance to the scheduling environment, is the idea of job dependencies. As the phrase implies, dependencies is the ability of the scheduler to control the execution of various tasks by having them execute not just on a certain date, but also in a particular order and in conjunction with other tasks.

The simplest expression of a dependency is Job B must follow Job A, or Job B cannot execute unless Job A has completed. However, this simple concept can get very complex very quickly. Job B might have to wait for a particular file to arrive; it might have to wait for data to be input by a user; it might have to wait for a different job that uses the same database table that it accesses to complete; or it might be restricted from running during certain hours. When looking at the dependency capabilities of a product, it is important to have a clear idea of the types of dependencies that exist in the environment and to be able to quickly map those to the product being evaluated. If the product cannot create the needed calendar, run jobs in the correct order or to satisfy the right dependencies, it is not likely to meet core business needs and will quickly become unusable.

Event-Driven Processing

When job schedulers were first created, the clear methodology for managing jobs revolved around time schedules and the management of resource dependencies. Job schedulers were developed to run jobs at the right time and in the right order. While those requirements still exist, there are new requirements for managing jobs on a real-time, event-driven, or ad hoc basis. Instead of scheduling jobs based on date and time criteria, the job scheduler should be able to dynamically introduce jobs (and entire business processes) into the schedule in response to a business event such as the arrival of a specific file.

The adoption of Java and .NET for hosting applications has sparked a need to manage batch transactions that are distributed across multiple platforms to fulfill the complete business process. However, neither platform inherently possesses scheduling features. To support these highly distributed application platforms, an effective job scheduler must accommodate the needs of both near-real-time and batch tasks across a diverse set of infrastructure platforms.

With an event-driven workflow processing approach, the job management solution does not continually "ask" (by polling) if an event has occurred, but is instead "told" immediately by communicating directly with the operating system. There are hundreds of events that can be used to trigger jobs - changes in a database, email arrival, network events, system events (such as memory utilization or disk utilization), and third-party file insertion triggers, to name a few. Event-based scheduling provides a means of removing latency from batch business processes and allows IT to be more responsive to the evolving needs of the business by delivering complex workflows faster.

Additionally, if a job scheduler is well integrated into the environment (see the section on APIs), the scheduler can also process events coming from other systems management products such as network managers, or even built-in or custom applications.

To make it easy and practical to access events, a job scheduling vendor should also supply event adapters that support the major business event sources. These should include the following event types:

- Job events (for example, job completed abnormally, job running longer than expected)
- System events (for example, lost connection to scheduling agent)
- File events (for example, file arrival, file reaches a size threshold, file is stable for period)
- Email (for example, order received in sales inbox)
- Databases (for example, new row added to table, row in table modified)

In addition, there may be a need to create customized event sources to trigger actions from applications. For example: Start a batch process whenever the number of transactions exceeds some threshold or whenever the cumulative value of customer orders exceeds some value. To do this, the job scheduler should expose an interface that allows the applications to externally update custom scheduler variables. Then the job scheduler can watch until the variable value matches a condition that identifies an event of interest. When it matches, the job scheduler should have the ability to trigger the appropriate business process response.

Queues and Load Balancing

Job schedulers need to manage both the number and the mix of jobs that can run concurrently on a given system or processes running within a specific application. If too many jobs compete for the same resource, they can significantly impact processing throughput, and therefore timely completion.

What makes this facet of the job scheduling environment particularly difficult to solve is that not all jobs are created equal. Not all jobs have the same priority or resource requirements. In addition, there is a finite pool of individual resources, and resource intensive jobs can immediately deplete this pool, creating resource contention with other jobs and applications.

Enterprise job schedulers should provide a means of managing the number of simultaneously executing jobs by selecting eligible jobs to run based on job and queue priority. For example, a queue can be created that supports only lower-priority jobs. All jobs of a certain priority or below are placed in a queue that is set to a lower priority than a queue that manages more important workloads. A classic example is to place all user-submitted jobs in one queue for execution during off-peak times, while regularly scheduled production jobs are placed in a queue with a higher priority.

A robust scheduling product should also support methods of distributing large and complex jobs across a distributed computing landscape. By giving operations staff the flexibility to manage performance load within their job scheduling environment, enterprise job scheduling can maximize job throughput and optimize service levels.

Resource Management

Even when there is an available queue sequence for a job, there may not be adequate resources to service that specific job. The job scheduler also needs some mechanism to ensure that there are sufficient resources available to service a specific job.

Resource management provides the mechanism for controlling the execution of jobs based on the amount of resources they will consume. This prevents jobs from competing with each other for limited computing resources. For example, a job may require higher VPN bandwidth to a trading partner, access to concurrent database client licenses, and any number of other resource constraints.

Resource requirements can be defined in two ways. First, resource rules can be used to start a job only if all the required resources are available, minimizing the chance the job will fail or be delayed. Second, resource rules can be used to restrict the number of jobs that can run concurrently and that consume one or more of the same critical resources. For instance, you can set up a rule to allow only one job run to execute at a time to update a specific database table.

Resource management needs to support both virtual and physical resource models. Virtual resource management allows you to throttle and control workloads and jobs based on a logical resource that may not be tied to a physical system attribute. Physical resource management allows you to throttle and control workloads and jobs based on a physical system attribute, such as CPU availability.

Together, queues and resources provide the means of minimizing delays in processing, maximizing throughput, and managing resource consumption. Be sure that the scheduler you are considering can schedule jobs based both on queues and resource consumption and also that the solution can auto-discover and update resource information as needed.

Framework and Network Management Integration

Many companies today have implemented some type of event or systems management console. Examples include HP’s Operations Center and Microsoft’s SCOM products. These management frameworks provide a variety of features and functions, but in many cases are implemented to provide a single-console view of mission-critical events happening within the data center and beyond. These consoles are typically “management by exception” solutions. In other words, given the incredible number of events that occur within a medium- sized to large data center, operations staff want to view and manage only exceptional conditions - typically the most serious anomalies.

Because a large number of IT processes run in batch mode, it is critical that the scheduler integrate with the data center's chosen framework or events management console. Typically, this integration will be twofold. First, the integration should allow the scheduler to inform the console when there is some type of job failure using standards-based messaging protocols. Second, the integration should allow the system and network management tools to monitor the scheduler and its associated infrastructure. Although not as obvious, this integration gives the event management framework operator the ability to monitor the health of the scheduler itself. In this way, if the scheduler or one of its distributed components experiences an error, the management console should be able to report on any failures.

Security

It should be apparent that security is a vital requirement of any job scheduler. If a job scheduling product is in charge of running the mission-critical processes in the data center, clearly access security controls must be in place. In addition to controlling access to the scheduler itself, an administrator may also want to control access to individual features within the product by user or by group.

The primary security controls for any scheduling solution should be in its ability to authenticate and track users in the system. You can have a single source of authentication by connecting to a standard corporate employee directory database. This allows administrators to control access to the scheduling solution and streamlines the process of adding, removing, or changing user access. The scheduling solution should also be able to continuously revalidate user status against the directory service and make the necessary rights changes "on the fly" as changes are made within the directory. The tracking of what the user does within the scheduler is a logical benefit of any granular security authentication feature. Can the scheduler record the footsteps of each user as they interact with definitions, jobs, and schedules?

Workgroup-based security management is another critical capability because it simplifies security management by allowing administrators to specify security policies for each job or job group, as well as for events and actions owned by a group. In this model, users must be members of a workgroup in order to own a job, event, or action, and can be tracked by the scheduling software for audit and change management purposes.

Managing security access down to the individual user level should be a requirement because of the diversity of users, not all of whom need access to all scheduling functions. For example, the IT administration team is typically given broad access to the tool, but restricting their access to certain jobs or certain features within the scheduler may be necessary. Other personnel may have the authority to create certain business calendars or jobs but do not have the ability to run those jobs in production. In some corporations, end users are given limited access to the product so that they can monitor the progress of jobs they own through self-service portals.

The key when looking at a scheduler's security features is to look for ease-of-use and granularity. Ease-of-use is necessary for quick authorization changes for a given user or group of users in response to changing business needs. It is dangerously shortsighted to compromise an operation's security policies simply because it is deemed too difficult to implement a particular policy. Granularity, or refinement, is important because of the need to make only certain features of the product available to certain users. Granularity makes it possible to easily grant the types of user rights to those users who need them, and to restrict the systems and accounts that specific users can access, with the added ability to track the "who, what and when" down to the individual user level.

Audit Trails

Many people relate audit trails to security. While there is a strong connection, this is not the only benefit of audit trails. With audit trails in place, operations personnel can monitor and, when necessary, undo changes to the scheduling environment.

As an example, even authorized and well-intentioned users can make mistakes when modifying the production schedule. Audit trails provide the means to understand precisely what changes were made to the production environment and who made them. Given the rapidly changing requirements of the modern data center, it is imperative that all changes to the production environment be recorded.

A related topic is the concept of system or scheduling logs. While true audit trails are typically focused on what changes were made by whom and when, logs simply document the results of a particular execution of a job or schedule. Well-organized logs give the user a clear indication of exactly what workload ran, when it ran, and its results.

Reporting and Analytics

A full-featured workload automation solution should provide detailed critical- path analytics and historical reporting to business end users, IT executives, scheduling managers, and IT staff. Good reporting on the performance of workload automation environments and the SLAs that comprise business processes is an important evaluation factor.

Questions to ask are:

- Can the analytics tools give me accurate discovery of entire job streams?
- Do I have the ability to deliver real-time job-run information to business end users?
- Can I accurately predict when a job will complete and will I be alerted proactively if resources will affect my SLA delivery?

Rapid access to accurate job stream reports can support IT in detecting issues proactively so that service levels are not compromised. Insight provided by sophisticated analytics can also support the auditing and compliance process, as well as supply new strategies to help continuously improve business processes. Analytics that delivers predefined reports, as well as the ability to create custom reports, should ideally be a prerequisite. The analytics tools should gather the data needed by IT and business managers to troubleshoot, trend, and analyze job scheduling performance across the enterprise.

Alert Management and Auto Recovery

Because job schedulers perform such a vital role by automating critical business processes, they must be able to generate alerts when something unusual happens within the processing environment. Alert management should be flexible enough to handle a wide array of potential events. It should also be extensible so that it can identify and manage events of the user's choosing.

Much like auto-recovery features, alerts functionality should respond differently to different types of events. In fact, alert management and auto-recovery often need to work in conjunction with one another. In a typical scenario, a job might fail, which would then initiate some type of recovery action. At the same time, there may be a desire to send notification of the failure to a designated person. This notification might be an email message to a specific user, an XML page to a technician, or a message sent to a central management console. Additionally, this alert functionality should allow the acknowledgement of the alert so that the scheduler is informed when the appropriate person has received the alert.

As with other features, ease of use and flexibility are extremely important. Some products claim to deliver these types of alert capabilities, but a closer examination reveals that this is only possible if the user writes custom scripts, which ultimately defeat the purpose of having an automated scheduling solution.

Some failures are not meaningful enough to stop subsequent processing, while others will corrupt critical downstream processing and therefore supply inaccurate reports or lead to other problems. Therefore, the product should allow the job scheduling operators to create multiple types of recovery scenarios. For instance, in some cases it might be sufficient for the scheduler to simply stop processing altogether if an error is deemed severe enough. In other cases, it might be decided that when a specific type of error occurs, the schedule should back up a couple of steps and rerun those jobs. If these remediation steps are not sufficient, the user may run a series of related recovery actions such as restoring a database prior to attempting to run the jobs again.

The products being evaluated should be able to take distinctly different actions based on the relative severity of each problem encountered. At a minimum, the product selected should allow the user to stop processing, continue, or rerun processing and/or run a series of recovery actions before moving on to a subsequent step.

Ease of Use

Although this feature is sometimes difficult to describe, most users feel “they know it when they see it.” More accurately, they tend to know it when they experience it. This might be viewed as a “soft” requirement, but it should not be underestimated. Staff turnover is frequent, and often the most experienced people are not onsite at the precise moment when a failure occurs or the daily schedule needs modifying.

For most users, the familiar Internet browser interface is the most intuitive of all UIs. And it has the added benefit of simplifying remote access. Newer products have generally adhered to this type of interface, and, in some cases, have added additional graphical capabilities to further simplify usability. It is also important to note that tools with intuitive interfaces are not only easier to use but ultimately produce fewer user errors and enable faster recovery when errors do occur.

A key part of making a scheduling solution easy to use is to eliminate the need to create custom scripts to define jobs and job streams, job dependencies, alerts, events, and other scheduling tasks. Scripts may be an easy “quick fix” to an immediate problem, but they tend to live forever while awareness of who created and managed them diminishes over time; all of this gives rise to “fire drill” responses when errors occur.

A modern scheduling and batch business process automation solution should be able to deliver flexible and repeatable solutions for these tasks through the use of drop-down menus, point-and-click selections, and drag-and drop user techniques. Avoiding custom scripting can significantly improve the manageability and quality of service delivered by a job scheduler.

Ease of Installation and Configuration

Given the dynamic nature of the modern data center, with new hardware and applications being added almost continuously, it is important that a scheduling solution be simple to implement and reconfigure. Ideally, the CPE should be able to reside on standard server operating systems within the data center; UIs should be browser-based and require no client installation.

During an evaluation phase, it can be difficult to assess configuration and reconfiguration criteria, but we encourage potential buyers to observe at least the initial installation of the product. Can the initial installation be self-service? If not, can staff members at least follow along with the vendor to understand the installation process? How long did the implementation take? Hours? Days? Any product that takes more than a day to install a basic implementation will probably not be any easier to reconfigure when the operating environment changes, and hence is not a good candidate for anyone who anticipates changes to their environments.

Other Job Scheduling Requirements

Most of the job-scheduling requirements discussed in this document are considered core requirements for a modern, distributed scheduling solution. Some of these requirements have evolved over time, such as the need for a relational database and the need for graphical management and a Windows-based interface, while others, such as support for business calendars and dependencies, have remained a constant over time. At the same time, additional requirements have emerged recently in job scheduling. These new requirements include the following.

Self-Service Portals

Many IT organizations are handing over some controls of various aspects of the job scheduling environment to business unit managers, giving these managers self-service delivery of non-mission-critical jobs. Data center IT staff can set security access criteria for business unit operations staff to allow them to define and run specific business jobs and reports. A web-based user interface, easily designed using supplied web service APIs, adds a new dimension for the distribution of job scheduling run time users and when and what they can access. This gives the business end user an easy-to-use way to manage their job-run requests.

When evaluating self-service web portals, ask yourself if these granular processes and security measures make self-service a reality. Will the portal take stress off of the IT operations staff, or will managing a portal be more trouble than it is worth?

Operational and Business Visibility

Real-time visibility and control of the job scheduling environment is a key requirement for business operational success, but a historical view of the scheduling environment is also critical. Scheduling environments are dynamic - a well-tuned operation is a constantly moving target. Job volumes, execution times, and processing patterns change as the business evolves. It is mandatory to employ business intelligence (BI) platforms that provide reporting that is granular and that support operations forecasting. This enables staff to be more efficient and to concentrate on proactive tasks such as optimizing schedule and increasing service levels quality. Having an integrated BI solution that is easy to use, powerful, and extensible is a must for responding to the changing enterprise scheduling environment and to user needs. See the section on Integrated Reporting above for additional details.

Scheduling Big Data

Big Data has become the latest industry trend. It is driven by business' challenges with capturing and managing the amount, type or sources of incoming data into their data centers. Difficulties include capture, storage, search, sharing, analytics, and reporting. IT regularly encounters these issues in e-commerce, search, bio research, finance, and business intelligence solutions.

For the truly scalable workload automation solution, these issues should not matter. The enterprise-class solution should scale to millions of jobs and interact with the appropriate applications and databases. Question to ask prospective vendors:

- Does the solution automatically scale for resource availably?
- Can I schedule from/to multiple staging areas, FTP sites, databases, and new unstructured data manipulation engine environments like Hadoop?
- Can my workload vendor act as a trusted partner to help me solve the larger network, compute, and storage infrastructure impacts to my data center?

Conclusion

An enterprise job scheduler is a core component of an overall systems management strategy. With job scheduling and the automation of end-to-end workloads, enterprises attempt to proactively manage the flow of essential business processes.

The core features of job scheduling are well understood and have stood the test of time. At the same time, each vendor has different interpretations of these core features. Additionally, the market continues to evolve and requires new features, continuous innovation, and new application and data support. Each data center must continually evaluate their business processing strategy to determine their workload automation requirements and whether their current solution continues to meets their ever-changing requirements.

Enterprise Job Scheduler Evaluation Checklist

Enterprise Job Scheduler Evaluation Checklist

Introduction

When purchasing a job scheduling solution, you should gather as much information as possible to enable an objective comparison of competitive products' key capabilities. This includes seeking out customer references, analyst recommendations, and press articles to supplement materials collected directly from the suppliers of the products under review.

This document provides a detailed list of evaluation criteria that you can use to benchmark the features and functions of the job schedulers that your organization is considering. The checklist provides a way to thoroughly assess how well a given product can meet the needs of your enterprise going forward. Of course, this is merely a starting point. Any requirements that are highly specific or critical to your operating environment should be added to the list.

Important Considerations and Checklist

In the complex distributed computing enterprises commonplace today, it is difficult to overestimate the crucial role that job scheduling plays in keeping mission-critical data flowing efficiently. While much attention is focused on high-profile business applications that are visible to a company's end users, the technologies that allow them to execute as designed, seamlessly and almost instantaneously, are the foundation for successful IT operations.

Given that it is routine for tens of thousands of jobs to pass through a data center daily, a flexible, scalable job scheduler is more indispensable than ever. When evaluating scheduling solutions for your enterprise, be certain that you have complete confidence in both the product and the vendor that you choose. Your due diligence in making this purchase decision will be rewarded with peace of mind.

Company Overview

- Financially stable company with global presence
- Strong technology reputation
- Expertise in data center automation
- Focused on the development, sales, and support of data center automation solutions
- Continuous innovation
- Customer support is 24/7/365
- Recognized by the market in the data center automation space

General Product Functionality

- Proven record of delivering on IT automation and business delivery goals
- Recognized for product ease of use and support
- Job definitions that do not require developer knowledge to configure or support
- Initial installation, configuration and job execution that can be completed in three hours
- Intuitive web browser interface for ease of use and error-free operation
- N-tiered architecture for maximum scalability
- Multiplatform support: for example, Windows, UNIX, Linux, OpenVMS, OS/400, z/OS, and HP NonStop

- Proven systems design for easy maintenance, low cost, and quick implementation
- Fault-tolerant architecture for automated system failover and uninterrupted scheduling, with a cluster-agnostic engine/master support for clustered high-availability environments
- Uses relational database to help ensure performance, reliability, security, and ease of reporting
- Scalable to hundreds of thousands of jobs per master, with thousands of agent connections
- Product supports inter master dependencies for multiple data centers or geographic locations
- Easy definition, execution, and management of FTP/SFTP/FTPS transactions
- Remote monitoring and management using wireless devices
- Agentless scheduling options for job execution on systems where scheduling software cannot be installed
- Extensibility to include scheduling of jobs on business applications and data center management solutions
- Extensible to include scheduling on custom developed applications

Enterprise Application Support

- Product has seamless integration with major applications such as Actuate, Amisys, Baan, Business Objects, SAP Business Warehouse, Cognos, Informatica PowerCenter, Lawson, Oracle eBusiness Suite, PeopleSoft, JD Edwards EnterpriseOne/World, SAP, Tivoli Storage Manager, and NetBackup
- Uses vendor approved integration points, SDKs, and APIs
- Supports other applications such as business intelligence (BI), extract, transform, load (ETL), data warehousing, database scheduling, backup, and so on
- Supports drop-down lists of available jobs, agents, variables, and parameters
- Allows for efficient means to transfer job definitions from development and user acceptance testing (DEV/UAT) environments to production system
- Integration adapters can be installed and activated in one hour
- Supports an application-specific user interface that saves rekeying of jobs already defined
- Application adapter does not use a command line with lengthy commands passing multiple parameters
- Integration does not require scripting language, but scripting is available for meeting complex, custom requirements
- Provides support of custom applications through API and command-line integration
- Provides web services interface for custom extensions to be created, such as for self-service portals
- Provides support for scheduling web services environments, which include Simple Object Access Protocol (SOAP)-based and representational state transfer (REST)-based methods

Date- and Time-Based Scheduling

- Schedules jobs by days, weeks, dates, times, intervals, and events
- Dozens of predefined business calendars
- Schedules jobs using a custom fiscal calendar interface
- Schedules jobs based on other job statuses such as Abnormal, Normal, Error Occurred, and so on
- Scheduling can be managed through command-line interface (CLI) and API
- Supports nested job groups for inheriting job characteristics
- Supports workload balancing with multiple options for distributing workload

- Schedules based on exit codes and exit code ranges defined by the user
- Schedules jobs and dependencies by parsing output for specific character strings
- Stores estimated, actual, minimum, and maximum runtimes for each job
- Granular scheduling to account for multiple global time zone idiosyncrasies

Event-Driven Scheduling

- Schedules based on file arrival, file existence, file size, and file stability
- Schedules based on database events
- Schedules based on changing variables (static, dynamic, and user-initiated)
- Schedules based on email events

Job Execution/Recovery

- Automatically restarts jobs based on job error or system failure
- Automatically inserts recovery jobs into the schedule for execution
- Supports multiple queues, job priorities, and queue priorities that include priority bumping
- User-defined parameters for job history information
- Supports ad-hoc execution of jobs as well as predefined schedule
- Supports operator overrides for parameters and dependencies
- Resource management to support both virtual and physical resource models

Security

- Flexible and easy to administer access rights
- Dynamic access and ownership rights validation
- Integrated with standard employee directory services for robust user authentication and group management
- Predefined and customizable security policies, including role-based security using existing user IDs and passwords
- Password encryption for all agent and application user IDs
- Administrators can grant/deny granular access to jobs, agents, and user abilities and enable view-only or operator-only consoles
- Secure Sockets Layer (SSL) data communications between solution components
- Supports SFTP to avoid sending data, passwords, and FTP commands in the clear, or unsecure manner
- Encrypted communications between master and agents

Auditing

- Extensive logging that includes all user interactions, job run statuses, and error messages
- Ability to set error, audit, and diagnostic levels for complete error handling and resolution

Reporting and Analytics

- Out-of-the-box reporting capabilities that deliver information about the entire scheduling environment from a single pane of glass
- Granular historical and real-time visibility and analytics into the entire job-stream critical path

Alert Management

- Comes with predefined system alert and job event triggers for notification
- Alert notifications including email, text, paging, console, Simple Network Management Protocol (SNMP), Windows Management Instrumentation (WMI), logging, and variable substitution
- Capable of including multiple job variables on any notification action, including job name, agent name, job output, actual runtime, exit code, and pager/cell number
- Management framework integration with HP Operation Center, SCOM, NetIQ, and others, using two-way communications for alerting and interaction

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

Printed in USA C07-704567-00 04/12