ThinkControl Architecture:

A Technical White Paper

think dynamics

Table of Contents

1. Introduction

ThinkControl is an automated resource management solution for corporate and Internet data centers, which focuses on the dynamic management of servers and network devices.

ThinkControl works in multi-application environments, proactively configuring servers and network devices to balance end-user traffic demands, excess capacity and specified service-level objectives. Using adaptive control technology, the system predicts capacity fluctuations, facilitates dynamic infrastructure reallocation and can automate an organization's best practices for managing data center resources.

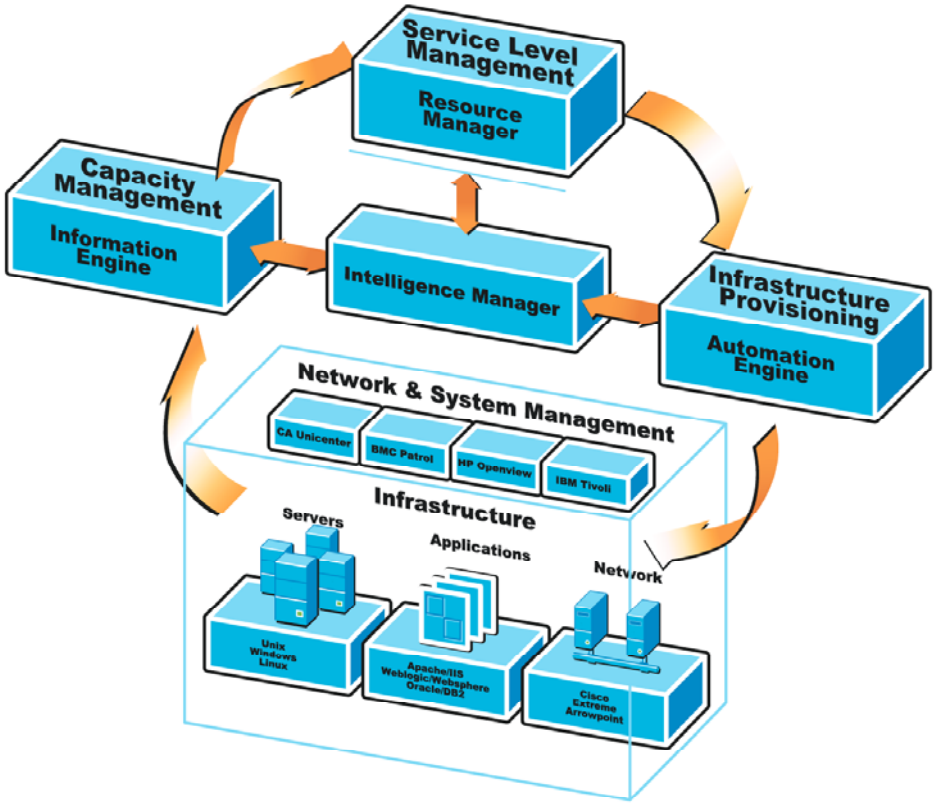
Each product module in the **ThinkControl** application suite (**ThinkUtility**, **ThinkAssure**, **ThinkRecovery** and **ThinkProvision**) is based on a common architecture. Each is a J2EE-compliant, Java-based, n-tier application that is fully scalable and can be distributed across multiple servers (Windows 2000 or UNIX). Each product also enables the creation and automatic scaling of n-tier, clustered application infrastructure including:

- Web server clusters, such as Apache, IIS, iPlanet web servers clustered with hardware or software load balancers
- Application server clusters, such as BEA Weblogic, IBM Websphere, Microsoft or open source
- Database servers, such as Oracle, Microsoft SQL Server, IBM DB2 and others

With **ThinkControl,** networking equipment may be automatically configured and managed, including load balancers and switches from Cisco, Extreme, Alteon, and Big-IP. Furthermore, **ThinkControl** offers heterogeneous support for Solaris, AIX, HP-UX, Linux, and Windows 2000 servers.

The **ThinkControl** architecture consists of four main software modules:

- *Information Engine*, providing real-time capacity management
- *Resource Manager*, enabling policy-based computing and service-level management
- *Automation Engine*, enabling the automated deployment of data center resources
- *Intelligence Manager*, storing data about the physical and logical assets of the data center



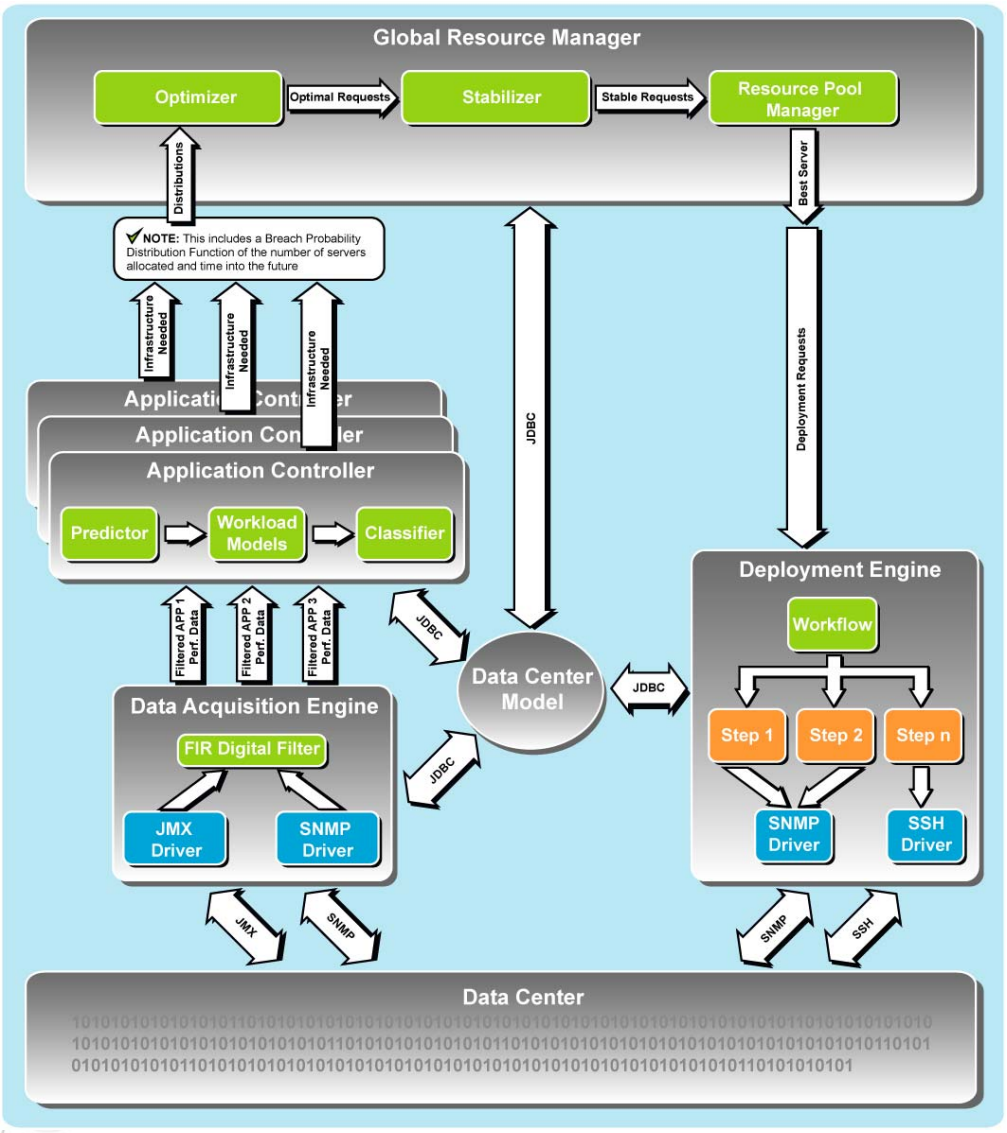
For a high-level description of these four components, please refer to the *ThinkControl Product Brief* (August 2002).

2. ThinkControl Architecture – A Closer Look

Within the four main components of the **ThinkControl** architecture described above, there are a number of key software modules:

- *Data Acquisition Engine* and *Application Controller* (parts of the *Information Engine*)
- *Global Resource Manager* (part of the *Resource Manager*)
- *Deployment Engine* (part of the *Automation Engine*)
- *Data Center Model* (central to the *Intelligence Engine*)
- *Management Interface*

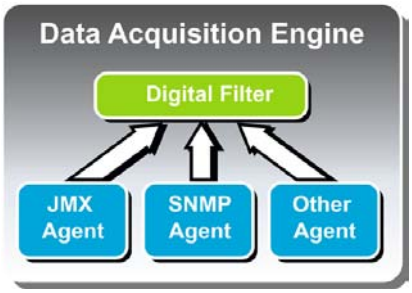
The figure below illustrates **ThinkControl**'s general architecture and shows how the main components of the system interact.



3. Information Engine

3.1 Data Acquisition Engine

This component acquires and pre-processes performance data from each managed application environment. Using a subscribing mechanism, it distributes signals to other components and filters raw signals.



The *Data Acquisition Engine* contains four acquisition components that gather information from their respective application environment layers:

3.1.1 Application Layer

This component gathers information from the application layer and all of the web, application and database servers in the system. This information might include the processing speed of requests in an application environment and the response time to such requests.

3.1.2 Operating System

This component gathers information from the operating system layer. The information collected here typically includes data that is used by both the infrastructure and application layers to expose performance data.

3.1.3 Server Infrastructure

This component gathers information from the server infrastructure layer. This information reveals how much of the total processing power is currently being used or how much of the total memory is currently being used. This information is based on the server groups that have been allocated to a specific application environment.

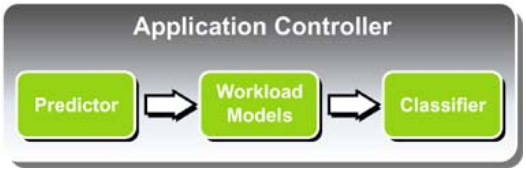
3.1.4 Networking Infrastructure

This component gathers information from the networking infrastructure layer through its *networking infrastructure interface*. This information is obtained from switches, routers, firewalls, and load balancers and shows how much of the bandwidth allocated to an application environment is being used and the transaction rates at a protocol level.

A *timing subsystem* controls the timing of the data acquisition and keeps a record of the last set of data obtained from each layer in each application environment. At predetermined intervals, the timing subsystem informs the acquisition controller that a set time period has elapsed. In response, the acquisition controller sends a command to one of the acquisition components to obtain data from a specific application environment. Data acquisition from each layer in the application environment can be performed simultaneously or can be staggered.

3.2 Application Controller

Application Controllers are created for each application environment under management. The *Application Controller* determines the resource requirements of an application using the real-time performance data provided by the *Data Acquisition Engine* and its workload models and predictions. The resource requirements are then sent to the *Resource Broker* component, which manages the overall optimization. The *Application Controller* incorporates prediction into an adaptive controller that recommends resource requirements to the *Resource Broker*.



The *Application Controller* contains components to monitor application performance and to predict future demand levels for the application environment, so as to maintain the predetermined level of service:

- *Prediction* component
- *Workload Modeling* component
- *Classification* component

3.2.1 Prediction

This component receives information regarding demand from the *Data Acquisition Engine* through the monitoring data interface and predicts future demand for the system's resources -- e.g. the arrival rate (hits/sec.) measurements for a web cluster.

The *Predictor* uses demand information from a particular server or network device to determine stationary (time-serial) and non-stationary (time-varying) trends. Stationary trends in the demand information tend to be random, while non-stationary ones are periodic, occurring at regular intervals, such as every day at a particular time, a particular day of the week, etc.

3.2.2 Workload Modeling

This component estimates each application environment's response to the incoming traffic. This involves obtaining the predicted future demand load from the *Prediction* component and gathering current performance information from the *Data Acquisition Engine*. The performance information consists of data that describes the current demand loads of a particular application environment as well as the environment's server performance data. If, for example, the performance data measures the utilization of the servers allocated to a specific application environment, this information can be used to assess the changing levels of demand and to determine the appropriate number of servers for that environment.

Based on this type of predicted demand, the *Workload Modeling* component estimates how the application environment will perform under the future, using a linear regression model. The performance and demand parameters that are determined include: service time, service rate and variance.

The *Workload Modeling* component also generates statistics for the critical CPU and arrival rates, which are then used to calculate the probability breach.

3.2.3 Classification

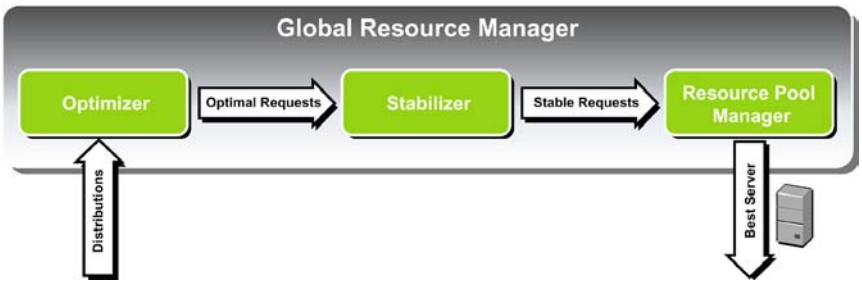
This component uses information obtained by the *Workload Modeling* component to determine how each application environment meets its predetermined service levels and the changes required to maintain them. A requirements detection subsystem determines the server demand for the application environment to maintain a predetermined service level. This subsystem uses real environment performance data that indicates which servers are being used; the level of demand; and application environment performance under these conditions. This performance data enables the server requirements detection subsystem to determine the operating requirements of the application environment. Similarly, the server requirements for the application environment are extrapolated from the current performance.

A degradation prediction subsystem predicts the way the application environment performance will deteriorate if the determined server requirements are not implemented. This subsystem compares the predicted degradation of performance with the predetermined service level for the application environment to identify any discrepancies.

4. Resource Manager

4.1 Global Resource Manager

This component receives requirements for servers or network devices from all the *Application Controllers* and manages their overall optimization. It has two primary responsibilities: to make server allocation decisions for optimal performance and to ensure that the application infrastructure is stable.



The *Global Resource Manager* considers the different server requirements for each application environment and then determines where the servers should be allocated. The server reconfiguration and allocation data is then passed to the *Deployment Engine*, which breaks down the configuration changes into commands that are formatted and sent to the servers and network devices involved to implement the changes.

The *Global Resource Manager* is comprised of four components that work together to determine the configuration and allocation for each server in the system:

- *Resource Broker* component
- *Optimization* component
- *Stabilization* component
- *Resource Pool Manager* component

The *Resource Broker* uses real-time optimization algorithms and manages the data center servers overall to meet individual system demands. The *Optimization* component receives the requirements for servers and network devices from the *Application Controller* and then determines the server configuration that best meets the anticipated needs of the application environment.

The new server configuration is analyzed by the *Stabilization* component to ensure that application environment requirements have been determined under stable conditions and that each environment will remain stable after the changes. Finally, after the new server configuration has been deemed stable, a *Resource Pool Manager* then determines the individual servers that must be added or removed from each application environment, in order to meet the anticipated demand changes.

4.1.1 Resource Broker

This component manages all of the resource pools and fulfills server requests from each of the *Application Controllers*. It also attempts to maintain some surplus servers for quick deployment into needy sites. When the overall demand on the data center is near its full capacity, this component weighs the needs of individual sites with their particular service-level objectives to determine which sites receive additional capacity and which sites are left underpowered.

This central component manages all requests to optimize the use of available servers. When it makes a decision requiring a change to the infrastructure, the *Resource Broker* issues commands to the *Deployment Engine*.

4.1.2 Optimization

This component uses the anticipated server requirements provided by the *Application Controllers* to balance the demands of different application environments for the same servers. Anticipated requirements are received through a requirements interface, which passes this information to a request type-sorting component, which then sorts the requests according to the type of server requested. For example, the requests for an additional server with a Unix operating system are sorted from the requests for an additional server with a Windows operating system.

4.1.3 Stabilization

This component analyzes and filters the server changes indicated by the *Resource Broker* to determine whether similar types of changes were recently implemented. It maintains server allocation and configuration stability, preventing servers from being reconfigured and reallocated as a result of erratic changes in demand or the performance of an application environment. The *Stabilization* component can, for example, apply a time-based filter that prevents multiple opposing changes for a specified period of time.

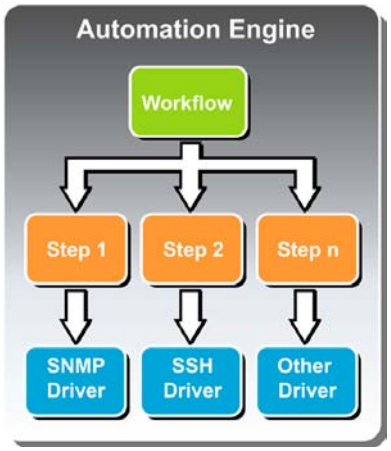
4.1.4 Resource Pool Manager

This component is responsible for the preliminary configuration and allocation of unallocated groups of servers. This process is important because it accelerates the actual allocation of servers to application environments. The *Resource Pool Manager* determines the most frequently used general configuration (for example, operating system) and then configures groups of unallocated servers accordingly, ensuring the availability of servers that have a frequently used configuration.

5. Automation Engine

This component is responsible for the creation, storage, and execution of repeatable workflows to automate server configuration and allocation processes.

Workflows play an important role in the *Automation Engine* design and consist of sequenced sets of commands. They can be large and complex or as simple as a single command. Both workflows and simple commands are considered commands. Since a workflow can be abstracted into a command, it can be included as a step in other workflows, therefore a workflow can contain simple commands, as well as other nested workflows. In this way, workflows make it possible to build a powerful library of processes that can be assembled to meet any data center process requirement.



Referring to the diagram above and in relation to the workflows, a driver is a Java class that contains the interface or the protocol code that will interact with the devices in the data center, including the **ThinkControl** service itself.

Simple commands serve as wrappers for drivers, describing a driver's input and output requirements through required and published variables. The simple command performs an action on the external environment, such as adding or removing a server, installing an application on a server, saving the configuration of a load balancer, etc. Drivers to external devices can be coded in SNMP, SSH, Telnet, or any scripting language that can communicate natively to the device.

The workflow framework and design of the drivers make it possible to manage any data center asset. Furthermore, **ThinkControl** provides full editing capabilities for workflow design. Existing workflows can be copied, referenced, or extended. This way, you can begin with a standardized process and then customize it to meet your organization's needs.

The *Automation Engine* supports a standard approach in creating system solutions. It enables the establishment of process standards, by offering the ability to use multiple references to the enterprise's standard workflows.

Changes to the data center environment made by the *Automation Engine* are reversible in two ways:

- By design, drivers have a *do* and *undo* method. While executing normally, the *do* methods are consistently called. The *undo* methods are called in the event of a failure, reversing the work performed
- By implementation, key workflows that build an environment can also have another workflow to break down that environment and return the components to a resource pool

Workflows enable users to create new application environments, create new application clusters, and offer the ability to add or remove servers to or from running application clusters. Workflows can also be created to perform any of the following functions:

- Deploy an application environment, or any part of it, on a server
- Reboot a server
- Configure network communications on a server
- Communicate with switching devices to reconfigure servers on a VLAN
- Communicate with load balancers to reconfigure a cluster
- Reconfigure an application environment
- Inform the fault management system of an invalid action
- Raise a billing event

6. Intelligence Manager

At the center of these continuous functions is the *Data Center Model* that circulates and maintains the information required to keep the data center fully optimized. This tool maintains knowledge of all physical assets, logical assets, customer accounts, application topologies, network topologies, service-level objectives, automated workflows, security policies and any manner of configuration. The *Intelligence Manager* is driven by the Oracle 8i Standard Edition DBMS, release 8.1.7 or higher.

7. Management Interface

ThinkControl's management interface includes two distinct user interfaces that can be used to manage and configure application resources:

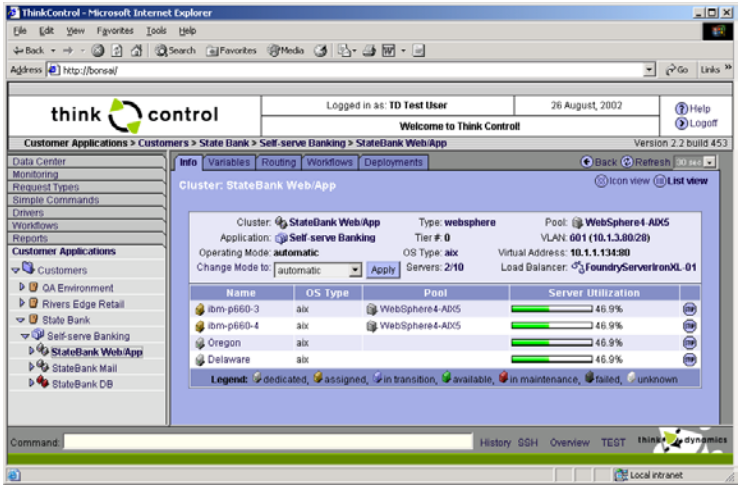
- *Web-based interface*: an intuitive way to display information about applications and components
- *Command line interface*: for operators who prefer to access the system's internal properties and operations using the command line

The management interface provides an overview of the state of all physical and logical assets in the data center infrastructure, offering information about the servers and their allocation and a means of generating configurations and allocations. It can also be used to create application environments.

The management interface interacts with the *Data Center Model* database to allow user access to its information. The user interface also communicates with the *Data Acquisition Engine*, the *Deployment Engine*, the *Application Controller*, and the *Global Resource Manager* components.

7.1 Web-based Interface

The web-based user interface offers a real-time access to **ThinkControl**, as well as an intuitive way to display information about the deployed application and components. Monitoring and controlling resource management and application performance is easy with it.



Using overlapped tabs on the left side of the interface window, the navigation trees allow the operators to easily configure and manage typical n-tier application architectures, as well as hardware assets and other system resources.

7.2 Command Line/Programmatic Interface

In addition to the web-based interface, **ThinkControl** also provides a command line interface for data center operators who prefer to access the system's internal properties and operations using the command line. It can also be used for creating scripts to automate repeated tasks. Certain internal properties and methods are available via a JMX interface through implemented Managed Beans (MBeans) to provide external applications with the ability to issue programmatic commands to the **ThinkControl** engine.

The following figure shows an example of the **ThinkControl** command line interface, illustrating the way a number of commands are implemented and executed. In this example, the operator is using the command line interface to change the default global operating mode to *manual*, changing the default operating mode for a certain application to *manual*, and is obtaining a list of all the servers available in a certain cluster.