Base One International Corporation
44 East 12th Street
New York, NY 10003
212-673-2544
info@boic.com
www.boic.com

Base One Grid Computing in Depth

Introduction

This document summarizes the features and components of Base One's distributed computing architecture, for harnessing the combined power of many Windows PCs and database servers to run large applications. The major benefits of this architecture are that it …

- makes it practical for groups of Windows computers to be applied to large-scale problems
- supports databases from all of the major vendors, including IBM, Microsoft, Oracle, Sybase, and others
- can be operated and administered either locally, on a LAN, or across the Internet
- incorporates a high degree of reliability, recoverability, and fault tolerance
- has extensive application security and administration facilities
- provides a comprehensive rapid application development (RAD) framework, enabling quick prototyping and flexible modification of existing applications
- includes comprehensive documentation, examples, and programming tools that make it easy to learn and use effectively
- generates efficient, scalable, multi-user applications that are easy to build, deploy, and maintain
- is capable of achieving timely, cost-effective solutions to demanding applications

For the sake of clarity, the underlying subsystems and layers of abstraction in Base One's architecture are presented from the bottom-up, in the following sections:

1. **Rich Client Core Components –** a RAD framework of class libraries and tools for constructing database applications that are efficient, robust, portable, and secure
2. **Distributed Batch Processing Services –** a general mechanism for automating repetitive, large-scale jobs in a distributed processing environment, built on top of Base One's core class libraries

3. **Grid Computing Middleware –** the layer that implements a “Virtual Supercomputer”, built upon Base One’s Distributed Batch Processing Services and core components

1. Rich Client Core Components

Under each Base One application is a core of general purpose programs (class libraries) and utilities for database application development. These facilities, comprising the Base One Foundation Component Library (BFC) and the Base One Internet Server (BIS), are implemented on top of Microsoft Foundation Classes (MFC) and .NET, enabling an organization to leverage in-house Windows programming skills, and assuring the smoothest, most complete integration with the Windows environment. Collectively, the modular components of BFC provide all of the essential ingredients of a complete RAD framework, tailored for maximum productivity with the full range of Windows operating system interfaces, including COM (ActiveX) and the .NET languages.

BFC’s class libraries are the basic building blocks from which Rich Client applications are constructed:

- Database classes – encapsulating all database storage and retrieval operations
- Screen control classes – elements for creating interactive user interfaces
- Reporting and Graphing classes – integration with Crystal Reports and Pinnacle Graphics
- Number class and Utility classes – components for high-precision arithmetic and a large variety of handy generic functions

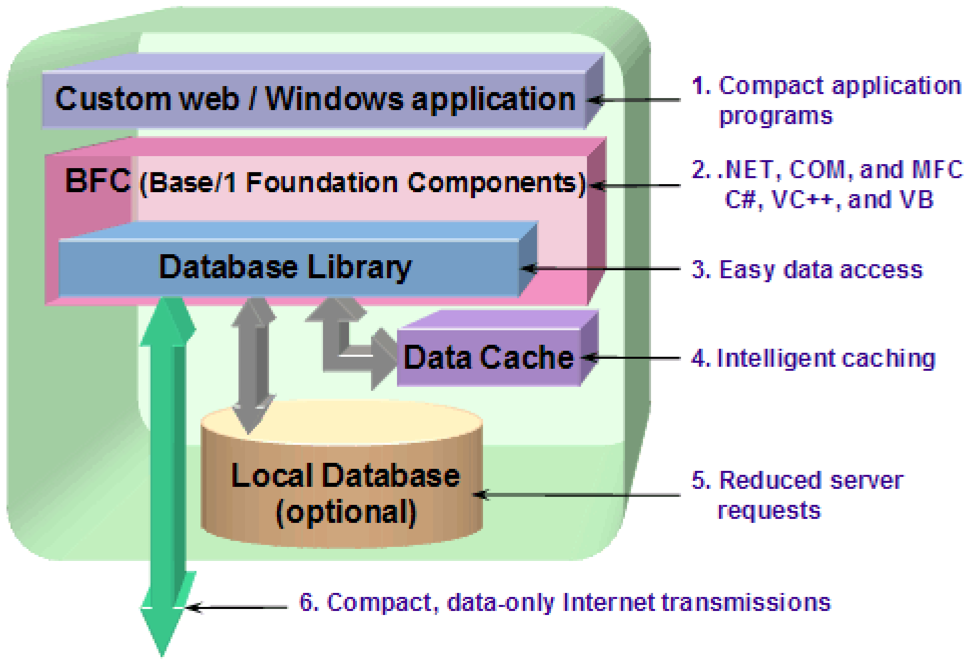
Accompanying the BFC class libraries are a number of closely related programs, which are integral to the complete Base One framework:

- Internet Server
- Data Dictionary
- Security System
- System Administration Facility
- Command Processor
- Database Trace Facility
- Help Creation Facility

Before elaborating further, it should be noted that there is a common feature in all of these programs:

> ***Comprehensive error and exception handling, with meaningful messages, and provisions for automated recovery from failures****. The full spectrum of errors, such as database errors (regardless of the database vendor), data entry errors, and disk crashes can all be logged and handled in a uniform fashion.*

Anatomy of a Rich Client Application


Custom web / Windows application
BFC (Base/1 Foundation Components)
Database Library
Data Cache
Local Database (optional)
1. Compact application programs
2. .NET, COM, and MFC C#, VC++, and VB
3. Easy data access
4. Intelligent caching
5. Reduced server requests
6. Compact, data-only Internet transmissions

1.1 Base One Database Classes

At the heart of Base One's design, with connections to virtually every other component, is the Database Class library. Its primary purpose is to make it easy to write efficient, reliable applications for large, multi-user databases. Also, the Database Classes provide a measure of portability, so that applications can be written once and used with any major database system, without reprogramming. Supported products include IBM DB2, Microsoft SQL Server and Access, Oracle, Sybase Adaptive Server, MySQL, and other networked databases.

Aside from the convenience of presenting a unified, portable database interface, Base One's software relieves programmers from some of the nastier details that are vital to database reliability and performance, especially as applications are scaled up. The Database Classes support these advanced features:

- **rigorous, efficient transaction processing**
  automatically synchronizing access and guaranteeing the ability to recover (rollback) from failures, while permitting a high degree of concurrency

- **intelligent use of local storage for database caching and buffering**
  increasing throughput by minimizing network traffic and server load

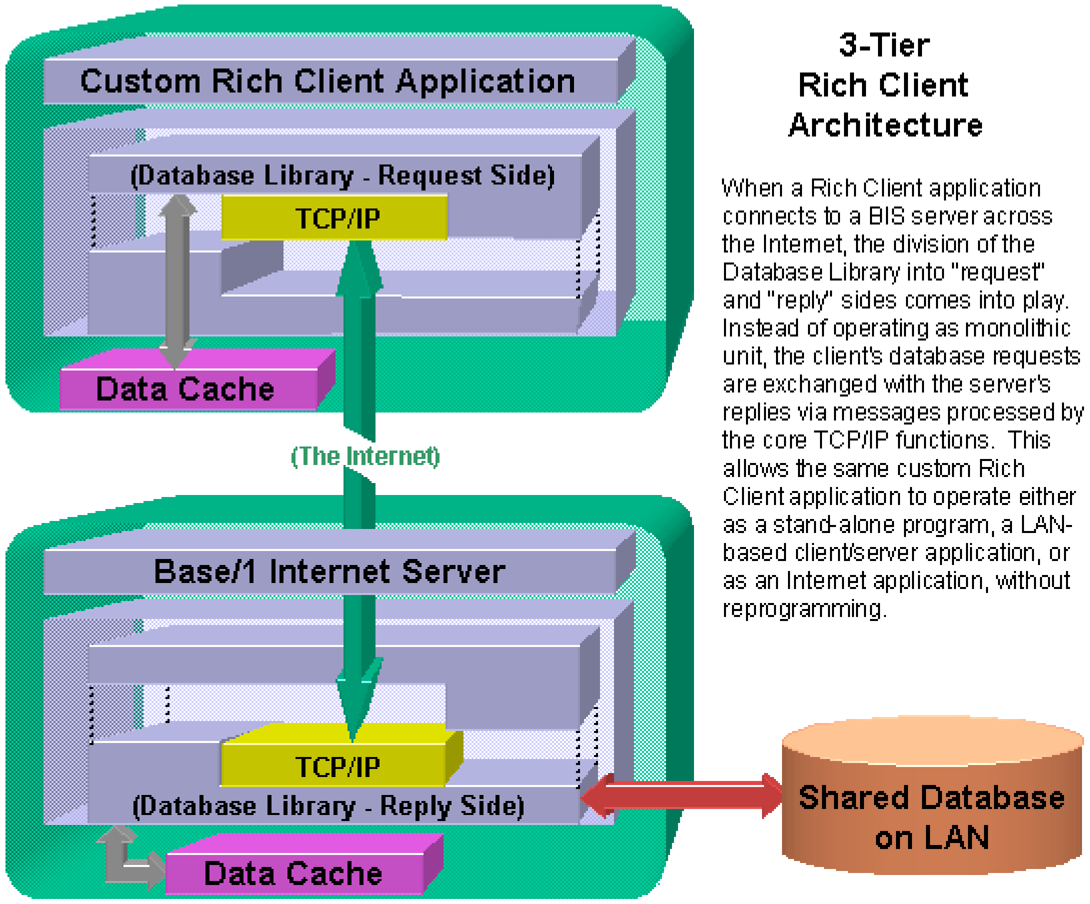
- **optimizations through high-efficiency, DBMS-specific APIs**
  using native direct connections instead of ODBC, as appropriate

- **multiple compressed BLOBS (Binary Large Objects)**
  extending individual records to include (logically) files such as graphics, text, sounds, Microsoft Office documents, or anything else required, as “attached objects”

- **interactive, script-driven database Command Processor**
  for local or remote administrative access to the server’s database

- **database trace facility for debugging and performance analysis**
  allowing you to trace through any BFC / MFC, COM or .NET application displaying or logging all database function calls, the values of passed parameters, SQL statements, and processing start and completion times.

1.2 Base One Internet Server (BIS)


Custom Rich Client Application
(Database Library - Request Side)
TCP/IP
Data Cache
(The Internet)
Base/1 Internet Server
TCP/IP
(Database Library - Reply Side)
Data Cache
Shared Database on LAN

3-Tier Rich Client Architecture

When a Rich Client application connects to a BIS server across the Internet, the division of the Database Library into "request" and "reply" sides comes into play. Instead of operating as monolithic unit, the client's database requests are exchanged with the server's replies via messages processed by the core TCP/IP functions. This allows the same custom Rich Client application to operate either as a stand-alone program, a LAN-based client/server application, or as an Internet application, without reprogramming.

The Base One Internet Server (BIS) makes it possible for the Database Classes to operate across the Internet, transparently, as if the data resided locally or on a LAN. Except for the Database Classes, programs don’t call BIS directly, so the complexities of Internet access are completely hidden from the rest of the system, and there is no need for additional network-related programming.

BIS moves data between remote databases and Windows clients (or web application servers). The database servers may be Windows, Linux/Unix, AS400, or OS390 computers. Using efficient, low-level TCP/IP communications, messages containing blocks of database and transaction processing requests and results are automatically sent to and from the Internet Server. As an added security measure, BIS is easily configurable for custom encryption of all Internet traffic.

1.3 Rich Client Applications

Programs built on top of BFC are “Rich Client” applications, which means they can operate more effiently in a Client/Server environment, using local client resources to reduce the load on the network and central database server. For example, it may be vastly more efficient to maintain local copies of lookup tables and selected files, or to perform local processing on intermediate result sets, rather than placing an additional burden on the server. The Rich Client model makes it much easier for applications to gain these kinds of performance advantages.

Since all database access in BFC goes through its Database Classes, any Rich Client application also enjoys the benefits of DBMS portability and network transparency. In addition, Rich Client applications can employ BFC’s integrated class libraries for Screen Controls, Reporting and Graphing, Numbers, and Utilities, as well as its administrative tools for Data Dictionary maintenance, Security, System Administration, Command Processor, etc.

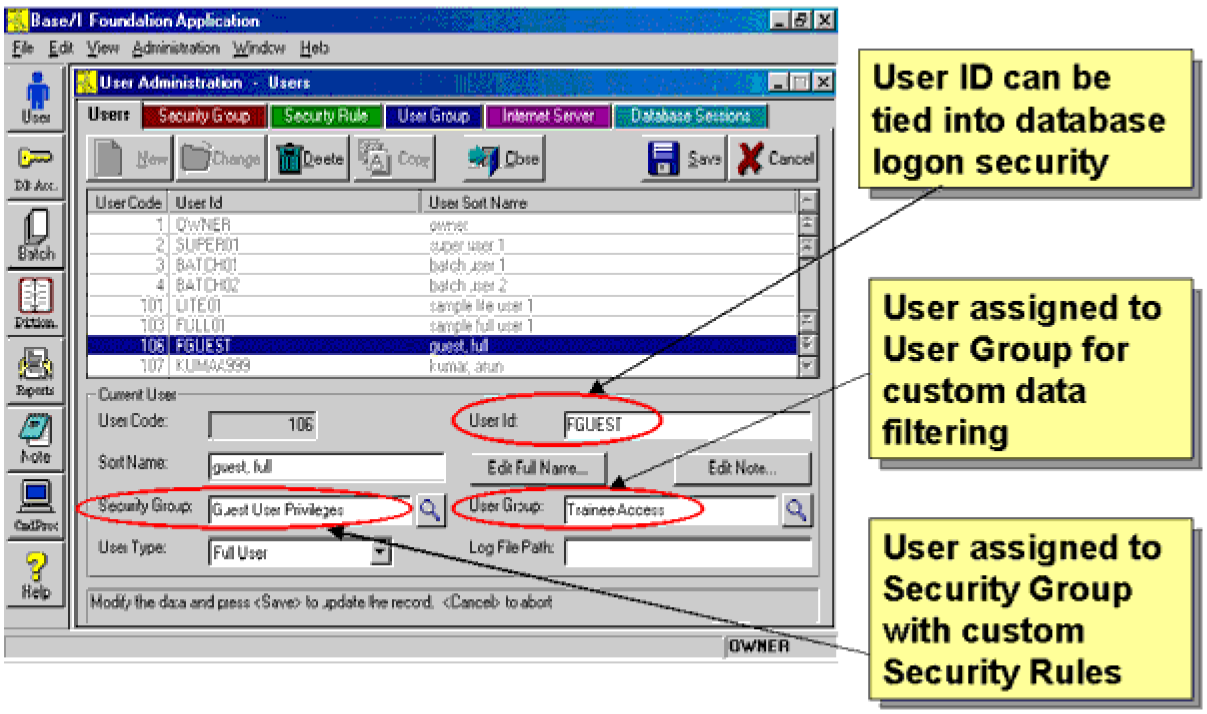
It's easy to build applications on the Rich Client model, because you only need standard Microsoft programming tools like C#, Visual C++, Visual Basic, VB.NET, ASP and ASP.NET. Rich Client relies only on Visual Studio, standard runtime libraries and COM/ActiveX components, interfaces familiar to many Windows programmer.

Many of BFC’s core facilities were themselves built as Rich Client applications, allowing administrators to perform complex server maintenance tasks without programming. This design has the virtue that it makes it easy to use these tools either locally or remotely, on a LAN, WAN, or across the Internet (through BIS). The programming behind these facilities also was greatly simplified by using the Rich Client architecture. Had we been constrained to develop a "conventional" browser-based implementation for our full set of server administration tools, there is no doubt that programming would have been far more difficult, and the resulting performance, reliability, and interfaces would have suffered.

1.4 Security System Components

Security is of particular importance in the context of grid computing, where a large, heterogeneous collection of PCs must be safely and reliably coordinated across the Internet. The Rich Client Security System has an extensive set of capabilities, tightly integrated into BFC's Data Dictionary, Screen Controls, and Database Classes. This provides a single, integrated authentication and authorization framework for all Rich Client applications.

In addition to fully supporting native database security features, Base One's Security System adds a number of significant extensions to provide finer control over users and application-level security restrictions. These DBMS-independent enhancements make it easier to build and administer highly secure custom applications, whether they run locally or across the Internet.


Base/1 Foundation Application
File Edit View Administration Window Help
User Administration - Users
Users
Security Group
Security Rule
User Group
Internet Server
Database Sessions
New
Change
Delete
Copy
Close
Save
Cancel
User
DB Acc.
Batch
Diction.
Reports
Note
CmdProc
Help
Edit Full Name...
Edit Note...
OWNER
User ID can be tied into database logon security
User assigned to User Group for custom data filtering
User assigned to Security Group with custom Security Rules

User Code	User Id	User Sort Name
1	OWNER	owner
2	SUPER01	super user 1
3	BATCH01	batch user 1
4	BATCH02	batch user 2
101	LITE01	sample lite user 1
103	FULL01	sample full user 1
106	FGUEST	guest, full
107	KUMAA999	kumar, arun

Current User

User Code:	106	User Id:	FGUEST
Sort Name:	guest, full		
Security Group:	Guest User Privileges	User Group:	Trainee Access
User Type:	Full User	Log File Path:	

Modify the data and press <Save> to update the record. <Cancel> to abort

The above is an example of one of BFC's User Administration screens. Through this and other such forms, the Rich Client Security System supports the following powerful security features:

- supporting **back-end database security** features for checking/changing passwords, grouping users and granting/revoking database privileges
- setting **per-application access privileges** on the basis of **user groups**
- defining **security groups** to control **access to a particular set of screen objects**, such as dialogs, forms, menus, buttons, web pages, and individual functions.
- specifying **prohibited actions** for a particular set of users through **security rules**

- **database session monitoring** – keeping track of **who is logged on**, using which application, **without cookies**, and regardless of what machine they're using

- restricting against **multiple logons** to the database by the same user

- establishing **custom encryption protocols** between clients and servers

- automatic generation of **per-user log files**

- **interactive facilities** for controlling database access, allowing administrators to **force users to log off**, **restrict logons**, and **broadcast messages**, for example, in preparation for system maintenance or shutdown

- built-in types of both **interactive and batch (non-interative) users** for efficient **sharing and pooling of server resources**, to handle very large numbers of users, distributed processing, and grid computing applications

2. Distributed Batch Processing Services

BFC includes the necessary components to build and administer very large, highly automated applications that efficiently harness the computing power of multiple PCs against a large, central database. Any Rich Client application can incorporate both the facilities for batch processing itself and for submitting, scheduling and monitoring batch jobs. BFC Basic includes evaluation copies of both Batch Job Servers and Base/1 Internet Servers, plus sample batch jobs.

In order to use batch processing, large jobs are broken down into smaller pieces that can be run individually, often in parallel, and with some set of interdependencies that determine constraints on the sequence job steps. For example, a large billing routine might be broken into sub-parts, each of which handles a different range of customer numbers. If multiple batch processors are available, the whole billing job could be sped up considerably. Once all of the billing steps are completed, another job might be automatically started to perform some consolidated accounting, and that might be followed by a reporting job, which may again be divided into a set of jobs that can run concurrently, and so on.

Since the Distributed Batch Processing Services are built on top of BFC and its Database Classes, batch programs automatically get the benefit of core features, such as simplified DBMS and network-independent programming, automatic transaction processing, caching, buffering, and local data optimizations, integrated security features, reporting and graphing capabilities, etc.

The Batch System's design also has these features and benefits:

- **robust database-driven architecture**, fault-tolerant because it relies only on transaction processing and integrity mechanisms, without the complication of fragile interprocess communication and master/slave relationships. Databases are fully protected from failures by well-established systems for reliability and recovery, as provided and supported by the DBMS vendors themselves, e.g. disk mirroring, multi-processor clusters, transaction logging, remote hot backups, etc.

- **supports a high degree of database concurrency**, ensuring scalability with a locking strategy that minimizes database lockouts, so that many users and batch PCs can share a common, dynamic database efficiently. (Programmers can employ basic transaction commits, without resorting to less efficient distributed transaction commit logic.)

- **automatically balances workload across a dynamic pool of available PCs**, yielding better overall thoughput by making fuller use of available computing power within an organization, or even outside of it (securely through BIS), freely intermixing low-end Windows PCs with high-end servers.

2.1 Batch Job Servers

A Batch Job Server is just a non-interactive Rich Client application that gets its work from a database instead of from an end-user sitting in front of the computer. Batch Job Servers look for work by examining Batch Job records retrieved from the database at regular intervals and each time a job completes. Once activated, the Batch Job Server decides which job to run next from the shared pool of pending batch jobs stored in the database.

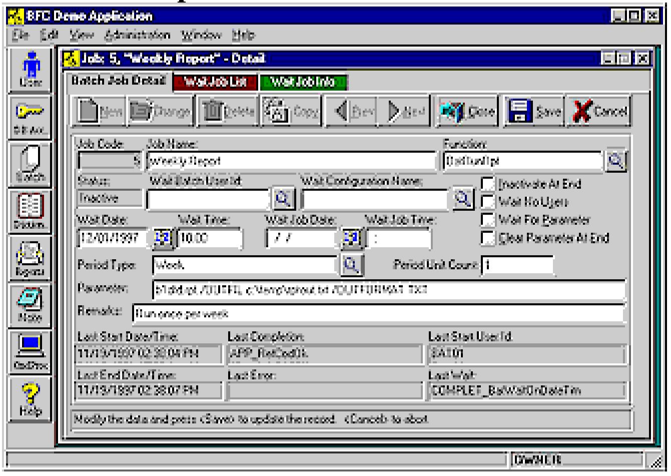
Any idle PC, whether on the Internet or a LAN, can be turned into a Batch Job Server. This is done by logging on to a Rich Client application using a Batch User ID. The computer automatically becomes a "batch machine" that is capable of running as a Batch Job Server. Batch Job Servers do not need to be configured as Windows servers. That is, a Batch Job Server can run under the inexpensive client versions of Microsoft Windows (such as Windows XP). If bottlenecks arise that would benefit more from increased horsepower per box, rather than further subdivision to a larger number of low-end PCs, then some percentage of Batch Job Servers are candidates for heavy-duty Windows server machines.

For every batch job there is a Batch Job record in the database containing the name of a Batch Function and the circumstances under which it may run. A Batch Function is just a C++ global function or a C# or VB.NET function in a .NET assembly, written according to some simple conventions that allow it to be associated with one or more Batch Job records. As long as end-user interaction is not needed, many existing DOS and Windows programs can also be run as batch jobs.

2.2 Batch Administration Facilities

Using a standard batch administration form, job dependencies can be set so that processing can be done in parallel by multiple machines. Batch jobs can be designed to wait for any combination of criteria, such as: a scheduled date/time or on completion of one or more other batch jobs with specific completion codes.

Sample Batch Job Detail Screen

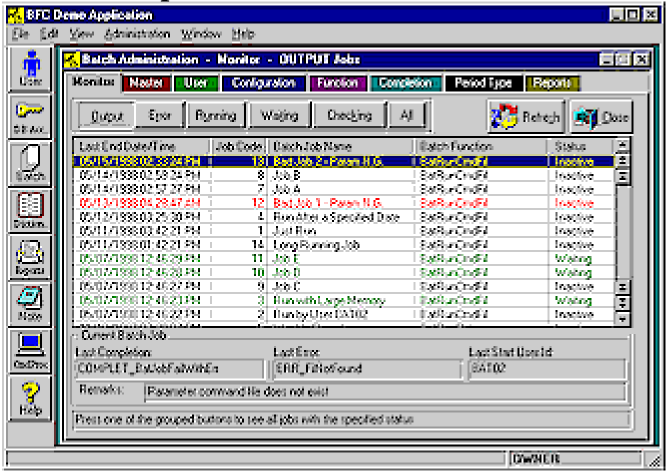


Job Code:	5
Job Name:	Weekly Report
Function:	BatRunRpt
Status:	Inactive
Wait Batch User Id:	
Wait Configuration Name:	
Wait Date:	12/01/1997
Wait Time:	10:00
Wait Job Date:	/ /
Wait Job Time:	:
Inactivate At End	☐
Wait No Users	☐
Wait For Parameter	☐
Clear Parameter At End	☐
Period Type:	Week
Period Unit Count:	1
Parameter:	b1dtd.rpt /OUTFIL c:\temp\rptout.txt /OUTFORMAT TXT
Remarks:	Run once per week
Last Start Date/Time:	11/19/1997 02:38:04 PM
Last Completion:	APP_RetCodOk
Last Start User Id:	BAT01
Last End Date/Time:	11/19/1997 02:38:07 PM
Last Error:	
Last Wait:	COMPLET_BatWaitOnDateTim

Modify the data and press <Save> to update the record. <Cancel> to abort.

The overseeing operators monitor the progress of batch jobs from their own workstations, logging on with their own User IDs. With the use of Base One's Internet Server (BIS), operations staff can launch jobs and monitor results from remote locations - while getting the performance benefits of running the jobs on batch machines close to the database.

Sample Batch Job Monitor Screen



Last End Date/Time	Job Code	Batch Job Name	Batch Function	Status
05/15/1998 02:33:24 PM	13	Bad Job 2 - Param N.G.	BatRunCmdFil	Inactive
05/14/1998 02:58:24 PM	8	Job B	BatRunCmdFil	Inactive
05/14/1998 02:57:27 PM	7	Job A	BatRunCmdFil	Inactive
05/13/1998 04:28:47 AM	12	Bad Job 1 - Param N.G.	BatRunCmdFil	Inactive
05/12/1998 03:25:30 PM	4	Run After a Specified Date	BatRunCmdFil	Inactive
05/11/1998 03:42:21 PM	1	Just Run	BatRunCmdFil	Inactive
05/11/1998 01:42:21 PM	14	Long Running Job	BatRunCmdFil	Inactive
05/02/1998 12:45:29 PM	11	Job E	BatRunCmdFil	Waiting
05/02/1998 12:45:28 PM	10	Job D	BatRunCmdFil	Waiting
05/02/1998 12:46:27 PM	9	Job C	BatRunCmdFil	Inactive
05/02/1998 12:46:21 PM	3	Run with Large Memory	BatRunCmdFil	Waiting
05/02/1998 12:46:22 PM	2	Run by User BAT02	BatRunCmdFil	Inactive

Current Batch Job

Last Completion:	COMPLET_BatJobFailWithErr
Last Error:	ERR_FileNotFound
Last Start User Id:	BAT02
Remarks:	Parameter command file does not exist

Press one of the grouped buttons to see all jobs with the specified status

3. Grid Computing Middleware

Taking a collection of Batch Job Servers in combination with one or more Base One Internet Servers, one can easily construct a “Virtual Supercomputer”, capable of securely sharing data and computational resources, on demand, over a widely dispersed area. This essentially constitutes a ***RAD framework for grid computing***, particularly well suited to large database applications, because it builds upon the unique capabilities of BFC’s Rich Client model and core Database Classes. In contrast to other approaches to grid computing, Base One’s Virtual Supercomputer has these distinctive features and advantages:

- **exceptional support of Windows PCs**, ranging from Windows XP to high-end Windows servers. Using Microsoft’s premier developer toolset, programmers can employ the full range of Windows technologies, including MFC, COM/ActiveX, and .NET, as well as a variety of popular third-party add-ins.

- **highly scalable, fault-tolerant architecture**, because it builds on the efficient, database-centric Distributed Batch Processing Services, which use robust native DBMS transaction processing reliability and recovery features.

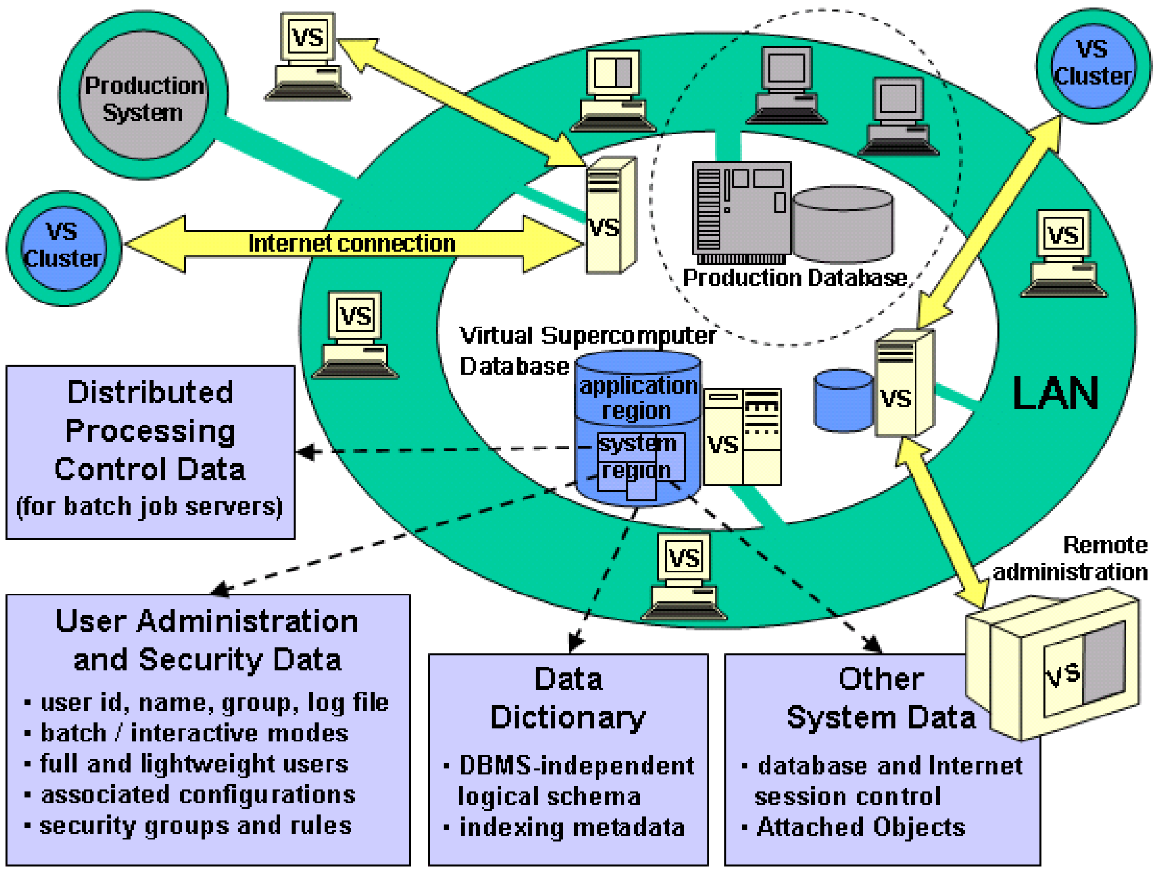
- **requires no specialized hardware or system software,** since Base One’s design is compatible with standard versions of Windows.

- **works with any major DBMS,** because this is a basic feature of Base One’s core Database Classes, including automatic support of high-speed native DBMS APIs.

- **provides extra security,** in addition to standard DBMS and operating system features, through the Rich Client Security System and BIS custom encryption.

- **enables rapid, controlled applications deployment,** because executables can can be installed and updated without touching the Windows registry, system directories, or other operating system files. There are no browser compatibility issues, since no browser is required. For client machines that make use of Base One’s Internet Server (BIS) for data access, there is no need to worry about ODBC settings or DBMS communication software compatibility, because the actual database software need only reside on these servers (BFC handles the rest).

The Base One Virtual Supercomputer

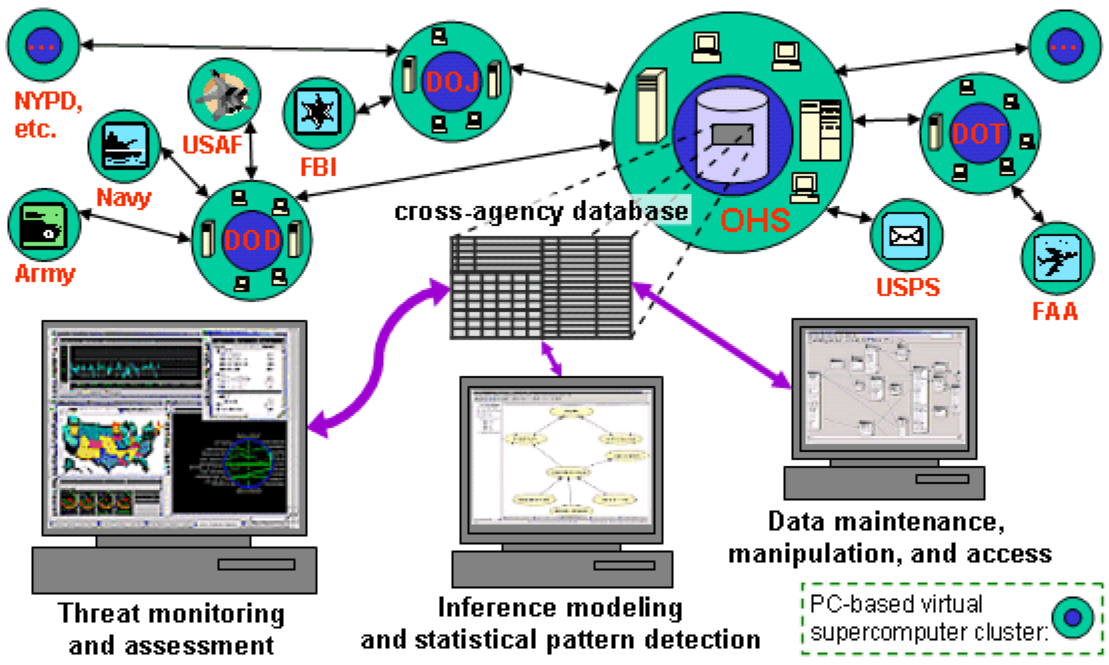

Production System
VS
VS Cluster
VS Cluster
Internet connection
VS
Production Database
VS
VS
Virtual Supercomputer Database
application region
system region
VS
VS
LAN
Remote administration
VS
VS
Distributed Processing Control Data (for batch job servers)
User Administration and Security Data
▪ user id, name, group, log file
▪ batch / interactive modes
▪ full and lightweight users
▪ associated configurations
▪ security groups and rules
Data Dictionary
▪ DBMS-independent logical schema
▪ indexing metadata
Other System Data
▪ database and Internet session control
▪ Attached Objects

The above diagram depicts a scenario where the production system (in gray) resides on an organization's LAN, in close proximity to a "data warehouse" managed by a Virtual Supercomputer (VS, in light yellow) on the same LAN. Some client PCs are dedicated to the production system, some to the VS data warehouse, and some PCs can access either system.

Through Base One's Internet Server (BIS), a large number of remote PCs (as well as other remote Virtual Supercomputers) may be making sporadic, sometimes heavy use of the data warehouse, with minimum impact on routine production activities. The VS can efficiently harvest production data at regular intervals, to maintain archival data, aggregate statistics, and other information suitable for querying, reporting, and sharing with a larger audience.

Taking it to the next level – a Virtual Supercomputer network

Much of the efficiency in Base One's distributed processing architecture derives from its database-centric design, which leverages the strengths of commercial database technology as to reliability, security, efficiency, and scalability. Capacity and performance can be maximized through a variety of techniques, such as high-speed transaction processing, buffering, caching, and use of low-level APIs, so that a single Virtual Supercomputer database could, in theory, grow to immense proportions simply by adding PCs and increasing database capacity. On the other hand, some very large applications would be better served by a loosely coupled network of multiple Virtual Supercomputers, each with its own database.


NYPD, etc.
USAF
FBI
DOJ
Navy
Army
DOD
OHS
DOT
USPS
FAA
cross-agency database
Threat monitoring and assessment
Inference modeling and statistical pattern detection
Data maintenance, manipulation, and access
PC-based virtual supercomputer cluster:

The above diagram illustrates such a scenario, carrying the idea even further to *clusters* of networked Virtual Supercomputers, each corresponding to a huge government agency, with its own infrastructure, bureaucracy, and regulatory restrictions. The sheer size and geographic scale of each agency warrants multiple, fairly autonomous sites, but each agency also requires some degree of central aggregation and control. Yet another level of aggregation is then required to bring together information from all of these sources into a single repository that can analyze, report, and act on the whole picture.

Appendix: Base One Grid Computing FAQ

What kind of applications are best suited to grid computing?

Traditionally, grid computing has been associated with computationally intensive scientific and numerical applications, but more recently there has been a growing interest in using grid technology for a wide range of large-scale commercial applications. The reason comes down to simple economics: you can build more powerful systems, more cheaply, by more efficient utilization of available computing resources.

Now that the benefits and practicality of this technology are becoming pretty well established, companies are taking a closer look, but often unsure how it relates to their own business. In rethinking application designs, they are discovering is that grid computing doesn't just apply to number-crunching, it also makes perfect sense for all kinds of database-intensive applications, the bread and butter of commercial data processing. The question no longer is whether grid processing is relevant, but where to begin.

Why is fault tolerance vital to the success of grid computing?

Comprehensive error handling and automated recovery are essential to robust, scalable systems. Grid computing especially pushes these limits of reliability, since it entails a large number of distributed components, all of which are subject to various types of unpredictable failures, such as computer hardware breakdowns, operating system crashes, loss of communications, and abnormal exits from application software.

Historically, it has been too difficult to program distributed systems that can automatically recover from failures in individual components and keep working. For businesses, this has meant that distributed computing by PCs was more of an academic exercise than a commercial reality. Large-scale, distributed computing is now practical, but only if the middleware has been designed to handle a variety of inevitable failures. Without fault tolerance, systems can't grow large or complex, which is just where grid computing is most needed and offers the greatest benefits.

What's wrong with master/slave grid architectures?

In a master/slave system, you still have all of the usual aspects of potential failure, plus the added complications and hazards of critical dependence on flawless inter-process communication. Base One's symmetrical architecture is simpler and more robust, because it eliminates this needless complication - by leveraging the substantial built-in synchronization, transaction processing, and security capabilities of the major commercial database systems.

Instead of depending on direct inter-process communication, Batch Job Servers simply perform database retrievals and updates, and Web Services are modeled as database operations. There is no need for complex additional logic or a special class of “master” computers, so any available Batch Job Server with suitable resources and permissions can do the work if another one fails, or is busy. Eliminating the master also removes a potential performance bottleneck, resulting in a system that can more easily be extended and dynamically adapt to changing workloads.

Is it safe to rely on extensive use of remote PCs?

It's safe to make maximum use of remote PC power because of Base One’s ability to absolutely control what can be executed on each machine, and where those programs can reside. (The only variables are when a pre-defined module may be executed and what data is to be passed to it.). You don't need crippling "sand boxing" restrictions to protect PCs, since the grid computing middleware only runs programs that it knows about, i.e. those that have been statically linked to the executable. Within a corporate network, the grid application can run efficiently from a secure file server, also making program maintenance more convenient.

This is entirely different from the less controlled browser applet model, where new programs can arrive from across the Internet and be executed at any time.

The Base One architecture does not transmit program applets (which, for example, must be prevented from accessing the filing system). Instead, grid computing applications use efficient compiled programs that are resident in advance and can safely take advantage of the Windows filing system for sorting, caching, temporary storage, etc.

Can Internet-based grid applications be made secure?

All of the underlying Internet traffic for grid computing passes through Base One's proprietary Internet Server, which easily can be configured to use custom encryption. Also the Rich Client Security System provides DBMS-independent application and user-level access control, automatically enforced through Base One’s core framework components.

In addition, BFC’s user administration can be fully integrated into the organization’s network security and the native security features of the back-end database system This further assures that grid applications can be thoroughly protected by multiple, independent methods of administering security.

How does Base One avoid a bottleneck at the central database?

Every DBMS has performance pitfalls, which lead even experienced programmers into designing applications that scale poorly or tend to produce deadlocks. Base One's Database Library, at the heart of its grid processing middleware, has been carefully optimized for each supported DBMS, to achieve reliable transaction processing with a minimum amount of database locking. This makes it easy to build efficient, scalable applications without falling into the usual subtle traps.

For example, a common mistake is the over-use of cumbersome, distributed transaction processing (XA) protocols in message-based architectures. The greater economy of Base One's approach is possible, because the Virtual Supercomputer relies upon a centralized database, even though processing power and temporary storage may be widely distributed.

Of course any database ultimately could be swamped by excessive load, but the capacity of a database server generally can be increased far more easily than upgrading the rest of the system. By reducing transaction processing overhead, and minimizing network traffic through intelligent caching, a single Virtual Supercomputer can efficiently support a very large number of interactive users and Batch Job processors. If that is not enough, one can build a loosely-coupled network of multiple Virtual Supercomputers, but this more typically would be motivated by business or organizational reasons, rather than any inherent database performance or capacity limits.

How is grid computing charged back to different cost centers?

The Base One architecture fits in well with standard usage monitoring and accounting software, making it possible to establish custom charge-backs for each grid application, without additional programming. What makes it easy is the way that the Batch Processing System has total control over which grid applications are permitted, on whose behalf, on which machines. Thus there is a straightforward way, using familiar, conventional accounting software, to ensure that expenses for centralized IT resources can be appropriately borne by the departments consuming those services. All that’s required is to associate different batch applications with different cost centers.

Any Batch Job Server machine can also perform processing on behalf of multiple Virtual Supercomputers, simply by running multiple batch applications simultaneously, each accessing a different set of Batch Job records. This sort of arrangement provides more flexibility in automatic charge-back accounting, still without programming. If a customized, application-specific charge-back scheme is desired, BFC's core framework and tools make the programming easy.

Does Base One's grid architecture work with .NET?

Efficient grid applications can be developed with Microsoft’s latest .NET suite of programming tools, C#, VB.NET, VC.NET, and ASP.NET, using Base One’s core components for database access, batch processing, data-aware ActiveX controls, and administrative tools. Sample programs included with BFC illustrate a variety of the possible combinations, such as .NET Windows Forms and Web Forms that use either ADO.NET or Base One’s database functions, or both. Unlike most other framework vendors, Base One is committed to a high level of support for *both* COM and .NET environments, providing the smoothest possible migration path from Visual Basic 6 and ASP to VB.NET, C#, and ASP.NET. Organizations heavily invested in VB6 and COM-based applications are not rushed into adopting the latest untested .NET technologies, because Base One’s flexible, proven framework allows them to make the transition gradually, according to their own business plans and requirements.

Base One's unified database and distributed computing interface not only eases the transition to .NET, it simplifies programming in any environment. For example, the code for adding, modifying and retrieving data looks the same, regardless of whether ...

- interactive or batch-oriented
- web or Windows application
- programming in C++, C#, VB, VB.NET, ASP or ASP.NET
- databases from IBM, Microsoft, Oracle, Sybase, or MySQL

These symmetries makes it easier to learn and use Base One's software to develop applications, and the results are more portable and easier to maintain.