Consolidating Sybase

This article explores a couple of basic Sybase consolidation techniques and the UNIX and Solaris technologies that support consolidation. It offers a couple of configuration tips on the way.

When consolidating multiple Sybase (or any RDBMS) hosts, designers have the option to implement either an aggregation solution, which would involve implementing each application as a separate database within a server, or alternatively co-locating multiple ASE server instances within a single operating system environment. The first of these techniques I refer to as aggregation, and the second as consolidation (or work load sharing). The former technique may involve developer time because the data & business models require reconciling. It also may lead to constraints for scarce resources within the Sybase instance. These are typically Sybase memory objects, or system database resources. While a choice between Aggregation & Consolidation exists, consolidation is easier and delivers more benefits.

Sybase is implemented in Solaris as a a number of processes that attach themselves to a single shared memory segment. (The processes are referred to as Sybase engines). The processes are linked in the process table and inherit a Sybase name from the master..sysservers table. (Actually at run time, the name is inherited from the -S switch in the run server file.) The data in master..sysservers table needs to be replicated into the interfaces file. (More recent versions of Sybase can utilise LDAP for this purpose). The purpose of the interfaces file is to map the Sybase server name onto a tcp/ip address consisting of {tcp/ip address:port no} Each data server requires two ports on the same system. The default Solaris syntax has been to document the Sybase name direct to a tcp/ip address, although using Solaris' name aliasing is syntactically supported and a superior method. As a diversion, you should always convert the generated addresses into hostname:port no format. Also Solaris will support multiple tcp/ip addresses for each network interface. It is thus possible to co-locate multiple Sybase names within a single instance of the operating system (without containers) and ensure that remote processes can find the correct database server without changing the applications code, or client configuration parameters.

The number of processes cable of running in a Solaris instance/domain is defined and controlled by the /etc/system file. It is folklore and good practice to derive the number of engines as either the same as the number of CPUs (or related in some way to the number of CPUs). There is no technical constraint in either Solaris or Sybase that mandates this tuning rule. In the world of workload sharing, the share of a domain or system utilised by a Sybase server instance needs to be actively managed and I recommend that the Solaris Resource Manager is used to perform this function. This is a superstructure product in Solaris 8 and integrated into the OS from Solaris 9 on.

The default memory configuration of Solaris systems is to implement Sybase's shared memory as 'intimate'. The effect this has is to 'pin' Sybase's buffer caches into real memory. This leads to the configuration rule that sum of the consolidated server's caches needs to be less than real memory. If this configuration rule is not implemented, it is likely that one (or more) Sybase ASE instances will fail to start. (Oh, by the way, always set your SHMMAX parameter to HIGH VALUES, it saves a reboot when you breach a bound constraint. Systems on sale today (64 bit systems), are generally configured with ample memory for this not to be a problem. Thirdly, SHMMAX is a permissive parameter, setting it higher than needed is free.) Also Sybase DBAs have historically chosen/had to

install their databases on raw disks and hence all the data cache is configured as database cache through the “Total Memory” parameter. (The UNIX file system cache is of no relevance to the BMS). Most systems administrators and solutions designers are also aware of his and 'reserve' memory for Sybase. In the world of consolidation the solutions designer needs to be sure that real memory is greater than the sum of “Total Memory”, plus the UNIX kernel image.

Sybase has a very rich semantic for abstracting disk objects into tables and rows, and this can be used to scale IO resource. I recommend that raw disks are presented by the systems administrator with permission bits set to 600 and then that file system links with relevant names are created. The `disk init` command should use the link name as the physname argument. This creates a level of indirection in the naming conventions and also permits a rich naming convention so that stored procedures such as `sp_helpdevice` can actually tell the DBA stuff about the disks in use. Discovering that the database is mounted on /dev/rdsk/md22 is not helpful!. The abstraction means that data can be moved between disks without changing the database configuration (although the database can't be running). The other huge advantage of this technique is that striping and locating the sybase devices across multiple disks, RAID devices, controllers or switches becomes transparent to the database's mounting script. It allows DBAs to begin to use the language of storage attribution, and leverage the system scalability.

The UNIX file system will permit multiple versions of Sybase to exist within a file system hierarchy. The default sybase installation model incorporates the Sybase version into the UFS directory name for the install tree. This will cope with the circumstances where the applications portfolio is running two (or maybe more) versions of the database. A further advantage is that effort to support patch management is reduced. Consolidation will force increased standardisation, which will lead to a reduction in the breadth of the problem as a reduced number of hardware platforms hosting sybase require to be patched and tested against diverse requirements.

In summary:-

- Decide to Aggregate or Consolidate, or how to combine these strategies
- Design your engine/process/instance map
- Design you memory implementation
- Design your disk map and its abstraction interface

2004-11-25 07:33:12.0