What Time Is It In Your Test Bed?

Understanding the Benefits of Clock Simulation

Jack Di Giacomo
TANDsoft, Inc.

Flash backward! It's 11:59 pm, December 31, 1999. The world is about to run amuck. When the clock strikes midnight, all applications not Y2K-compliant will reset to the year 1900. Planes will fall from the sky, nuclear missiles will mistakenly launch, and failed power grids will cause catastrophic blackouts. Worst of all, the pre-record functions of VCRs will cease operation.

As we now know, the world did not descend into chaos on January 1, 2000. Despite dire predictions of global computer meltdown, the Y2K bug quickly became an historical footnote. In the early days of computing, back in the 1950s, 1960s and 1970s, programmers abbreviated years in a two-digit format (4/5/68 instead of 4/5/1968) to save expensive, scarce storage space on punch cards and later disk. Even back then, some scientists began lobbying for a four-digit year in anticipation of the rollover to the new millennium. They were dismissed as alarmists; the millennium was a long way off. Not until 1993, when curious staff at NORAD (North American Aerospace Defense Command) turned their system clocks forward to January 1, 2000, and watched as the ICBM alert system crashed, did governments, industries and individuals begin to react seriously to the Y2K bug.

When they finally reacted, they responded en masse. The global campaign to address the "time bomb" required everyone to upgrade their systems for Y2K compliance. How to accomplish that and then test for success was the challenge. How could developers recreate the millennium rollover without changing the system clock and risking a crash? Out of that challenge grew a new industry, one providing *clock-simulation tools* that could feign the date of January 1, 2000, without affecting normal system operation.

Clock Simulation Has Value Today

The need for clock simulation did not disappear at the stroke of midnight on January 1, 2000. If anything, the Y2K problems found in practically every programming language exposed the shortsightedness of the computer industry in not developing applications with time sensitivity in mind. Some recent examples include:

- Y2K7 bug – caused by the Daylight Saving Time (DST) date change in 2007 and affected any device that automatically corrected its system clock to the earlier DST date.

- Z2K9 bug (or the Zune bug of 2008) – Leap Year was not kind to owners of Microsoft's 30-gigabyte Zune MP3 player. Zunes around the world crashed at midnight on December 31, 2008, due to a problem with the player's internal clock.

- Y2K38 bug (or Unix Millennium bug) – software and systems that store system time as a signed 32-bit integer will be affected in 2038.

Clock simulation today is valued by companies whose development, testing, quality assurance and other applications are carried out on the same system. Recently, the popular move to data consolidation has accelerated the practice.[1] The resulting dilemma is that multiple developers working on multiple applications need to test their programs before putting them into production. With only one system available, how to accomplish that in the limited time available presents a challenge. When the programs to be tested require different date/time specifications, how to certify these applications without constantly changing the system clock becomes more than problematic. The solution is clock simulation.

Traditional Testing Approach

Without clock simulation, an application must rely on the system clock; and all applications running on that system must use the same date/time configuration. In such an environment, the traditional approach to testing applications is to change the system date and time. That's risky, to say the least, and presents myriad and often unforeseen challenges. Here's a few:

- System clock changes affect *all* programs running on a system, not just the application being tested.

- Resetting the system clock increases the time required for project development. Once the system clock is reset to local time, a system reboot may be necessary, applications must be reloaded, and each application's proper operation must be verified. It may also require that all users be logged off before resetting takes place.

- Forgetting to restore the system clock to its original state after testing or some other activity is completed may deny other users access to the system and may prevent applications from running at all. Software licenses and passwords may immediately expire.

- Often, only one time-sensitive process can be tested at a time, further delaying an application's deployment.

[1] See my previous article, *Application Jet Lag – Consolidating Global Data Services*, in the May/June 2009 issue of The Connection.

- Imagine the complicated, labor-intensive coordination required between IT administrators and application-specific staff every time the system clock needs to be reset.

Benefits of Clock Simulation

With clock simulation, test, maintenance, and development protocols can be undertaken much more efficiently and will be less prone to errors. Applications already executing do not risk disruption from tests performed on other processes because normal system operation is unaffected.

- Multiple applications, including test suites, can be evaluated simultaneously and cost-effectively while coexisting on a consolidated system, with each application benefiting from use of its own virtual clock. Applications not involved in testing will continue to operate off the system clock.

- Third-party solutions and utilities can be tested thoroughly prior to incorporating them into stable production systems.

- Clock simulation offers a cost-effective alternative to the expensive capital investment in hardware, software licenses, and IT resources that would otherwise be required to implement duplicate testing environments.

- Production consistency in batch-processing applications can be verified by testing overnight runs for date continuity.

- Disaster-recovery efforts can be expedited by allowing last week's work to run at the same time as this week's work.

- Forward-test such unusual circumstances as Leap Year rollovers or month-end processing on public holidays.

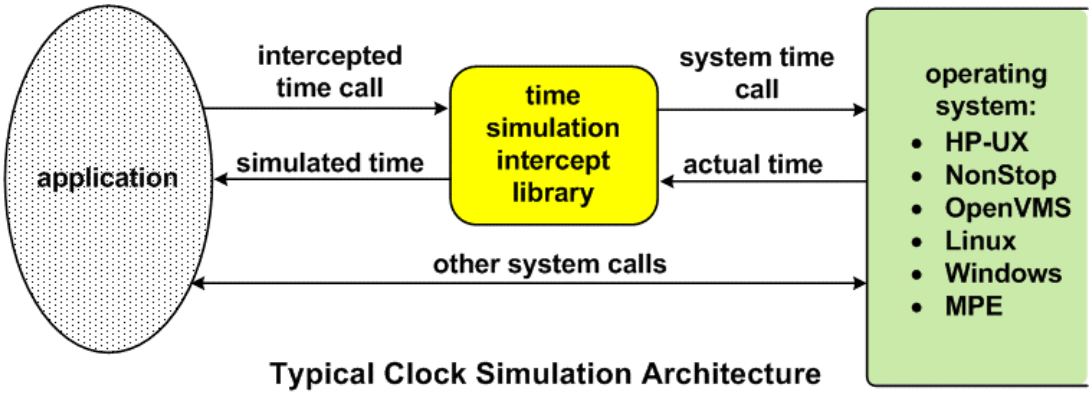
- Time-sensitive testing can be performed round-the-clock, not relegated to after-hours or weekends.

How Clock Simulation Works – A Representation

Whether you build an in-house product or purchase a third-party solution, clock simulation generally is based on the creation of intercept libraries. They provide virtual system times arbitrarily offset from current time, typically in the future but in some cases in the past. Services

are offered via application-groups, the definition of which can be flexible. Using grouping and wild cards, an application group can consist of a specified set of programs, a specified set of processes, a specified set of users, the descendants of a common ancestor, and so on. An application group can be defined to include only a single user or process. If an application is not defined within any application group, it continues to operate off the normal system time. A system can host many application groups, and each application group can specify its own time offset.

Via the intercept libraries, clock simulation is accomplished by intercepting system-time calls. No matter the application, the programming language, the system type, the program type, or the operating environment, all time calls ultimately generate a system call to a handful of operating-system procedures. They return a fixed or dynamic time in one or more formats, such as date/time or a Julian timestamp. Employing a command interface, a user can enter a date and time to be used by an application group. Thus, whenever a process within the application group makes a time call, the clock-simulation product will intercept it and will return a time that is appropriately offset from the system clock. Most third-party clock-simulation products require no application modifications to a system.



Typical Clock Simulation Architecture

Clock Simulation Products

The following companies offer date/time simulation tools for a variety of HP platforms.

1. OPTA2000™ (TANDsoft, Inc.) – **www.tandsoft.com**
   Clock simulation for HP NonStop servers. Offers a virtual date/time at any point in the past or future for development and test purposes. OPTA2000 requires no application modifications and supports virtually all NonStop environments, including S-series, Itanium, and Blades. OPTA2000 allows existing production, development, and backup systems to support worldwide consolidated environments. It eliminates the need to change system clocks in order to test time-sensitive environments.

2. Time Machine® (Solution-Soft) – **www.solution-soft.com**
   A time and date simulation tool that speeds application testing and deployment for HP 3000 MPE, Linux, Windows, and HP-UX servers. No application modifications are necessary, and there is no need to alter the system clock. Time Machine can simultaneously run up to 20,000 individually defined virtual clocks. A powerful application for testing "what if" scenarios on systems resources and programs.

3. HourGlass™ – (Allegro Consultants, Inc.) – **www.allegro.com**
   A date/time simulation tool that offers extensive support for testing batch and online application programs with future or past dates and times. HourGlass is available for HP e3000 (MPE/iX), HP 9000 servers, and HP Integrity Servers (HP-UX). Installation is simple and requires a single reboot.

4. DateWarp® (Vedant Incorporated) – **www.vedanthealth.com**
   A system clock simulator for the automated validation and testing of HIS (health information services) applications. A useful tool for adding realistic complexity to test designs, thereby minimizing the problems associated with resetting the real system clock. Available for OpenVMS and VAX operating systems.

Clock Simulation In Practice

- The User Acceptance Testing (UAT) group of a major enterprise comprises five test groups. Each group is responsible for ensuring the functionality of new applications or application upgrades before they are put into production. Applications include custom applications and third-party applications such as ACI's Base24. Each of the five groups had its own NonStop S-series server for acceptance testing and quality control. Recently, however, the company decided to consolidate the five servers into two NonStop Itanium servers to be shared by all five test groups. Many of the applications being tested are time-sensitive and must be tested under different date/time scenarios. The UAT group successfully employs clock simulation to facilitate acceptance testing of multiple applications running simultaneously on the same server but requiring different date/time environments.

- A global telecommunications company uses HP-UX servers for its Accounts Receivable (AR) development and testing environments. The company routinely needs to perform time-sensitive tests to certify that the AR system will perform accurately at the end of each month, each quarter, and each fiscal year. The company originally undertook the traditional approach of shutting down all time-sensitive processes, changing the system date for each process to be tested, testing the programs, reverting back to the system clock's original

time, and then restarting all the system applications. This process created a significant delay in the deployment timetable. After brief consideration of incurring the expense to implement multiple duplicate environments, the company chose a date/time simulation product that gives them the ability to run simultaneous time offsets within a single system.

- A prescription drug insurance provider manages both public and private plans for Canadian citizens. The company uses two S7400 NonStop servers for their applications, one server for production and the other for backup, development and testing. Prescription drug requirements change regularly, and the company must test a multitude of application updates in advance of placing them into production. For instance, a particular medication may cost a certain amount before September 1st but have a different price after September 1st. A generic drug unavailable before November 15th may come on the market after November 15th. A medication covered by insurance one month may not be covered the next month. Testing every change by altering the system clock is infeasible, so the company uses virtual clock simulation to forward-date the test protocols in order to identify any coding errors.

Less Than 7,991 Years to Go Until Y10K

Flash forward! It's 11:59 pm, December 31, 9999. The world is about to run amuck. Y2K's quick fix to four-digit year formats only lasted for 8,000 years - so here we are again. When the clock strikes midnight, all applications not Y10K-compliant will reset to 0000, or Year 0. Space colonies will fall out of orbit, neural networks will disintegrate, bionic limbs will cease to function, and even more VCRs will bite the dust.

Of course, it is the current and widespread belief that the year 10,000 is a long way off. Today's software designers, as did their colleagues of the mid-20th century, assume that their programs eventually will be replaced. Like cockroaches, however, software routines seem to possess a potentially infinite lifespan. Who truly knows what errant line of 21st century code will embed itself into 101st century technology.

For those of us planning to be around on that date via upcoming breakthroughs in life-extension, cryonics, or time travel, the Y10K bug is an impending reality. Some industries, such as those that examine proposals for the long-term handling of nuclear waste, already are developing modeling programs with five-digit formats.

You can bet that those programs are not being tested by resetting a system clock. What's making the difference in predicting the future? Clock simulation - it's all about feigning the time.