Concurrent driver upgrade: Method to eliminate scheduled system outages for new function releases

A. Muehlbach
B. D. Valentine
D. Immel
M. S. Bomar
T. V. Bolan

The reliability, availability, and serviceability goal for an IBM zSeries® system is to provide 24-hour-per-day, 365-day-per-year service with reduced system downtime. The continuous reliable operation rate has been improved with every new zSeries release by using error prevention, error detection, error recovery, and other methods that contribute to avoiding unplanned interruptions. Until now, planned downtime—for example, the time needed to upgrade a zSeries to the next firmware driver—had not yet been addressed. We developed the concurrent driver upgrade (CDU) feature so that customers could add new functions without downtime. It is now possible to upgrade the zSeries firmware engineering change (EC) driver to the next EC level without any performance impact during the upgrade. This paper describes the motivation and strategy of the CDU and describes its use.

Introduction

IBM zSeries* systems can activate firmware fixes concurrently[1] without the need for planned system downtime. Two different concurrent firmware fix strategies are used: *unit-concurrent* and *unit-disruptive*. *Unit-concurrent* means that the activation of a specific piece of firmware is concurrent from an application perspective. Technically speaking, when switching the firmware the system might be halted for a very short period of time, but it would go unnoticed by an application. *Unit-disruptive* means that the activation of a specific piece of firmware is concurrent for the system but is always disruptive for the subsystem. To make a unit-disruptive approach concurrent, either of the following is necessary:

- The subsystem exists more than once ($n + 1$ approach). One subsystem can then take over the system responsibilities while the other is updating its firmware.
- The subsystem is not required by any running operating system or application. The firmware can be updated at any time without affecting the system.

[1]Concurrency requires that the activation is transparent to any running operating system and has no impact on the operating systems and applications that are running.

History of the zSeries concurrent firmware fix feature

The concurrent firmware fix feature was introduced in 1994 with the implementation of the concurrently patchable code for the hardware management console (HMC) [1], followed by other zSeries components such as the support element (SE), the power code, I/O adapters (channel code), and flexible support processor (FSP).

Since 1995 it has been possible to apply System z* firmware fixes unit-concurrently. In the years following 1995, more subsystems implemented or improved the $n+1$ approach to use unit-disruptive techniques, or added or improved the existing unit-concurrent support. Details are provided in **Table 1**. Today, almost all firmware problems can be fixed concurrently. However, new functions cannot be added concurrently because of subsystem-specific limitations.

Motivation for CDU

Over the years, System z customers have increasingly accepted the concurrent firmware fix feature. Today, they consider it mandatory that any firmware fix is applied concurrently without planned system downtime. Consequently, customers are asking for a concurrent driver upgrade (CDU) feature that reduces their planned downtime.

©**Copyright** 2007 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the *Journal* reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to *republish* any other portion of this paper must be obtained from the Editor.

0018-8646/07/$5.00 © 2007 IBM

Table 1 Concurrent firmware fix history.

Firmware component	*Concurrent fix support since*	*System z model*
HMC	09/1994	9672 G1
SE, FSP, power, channel	07/1995	9672 G2
i390, processor unit, microcode, logical partition (LPAR), coupling facility control code (CFCC)	09/1996	9672 G3
Millicode	06/1997	9672 G4
Channel configuration off disruptive alternative	12/2000	2064
CFCC partition reactivation disruptive alternative	05/2004	2084

Customers have communicated several specific requests. They want no interruptions when either microcode fixes are applied or new functions are added using the firmware driver upgrade. Those without a Sysplex environment [2] do not want to switch to a Sysplex in order to be able to add new functions concurrently. Also, customers want the ability to decide when they will upgrade and add new functions. These requests can be met only if new functions are delivered with a new firmware driver and not within a sequential firmware fix stream.

Technical basis of CDU

Because of a strict IBM System z9* time-to-market requirement and the need to reduce the risk for unplanned incident repair actions (UIRAs), it was decided to reuse the existing concurrent fix capability of the zSeries for the CDU feature. The deciding factors included the facts that the existing firmware did not require major restructuring and that the risk of a UIRA is reduced by the continual improvement of the concurrent firmware fix feature.

A number of requirements had to be met to include the concurrent firmware fix feature in CDU: The existing concurrent firmware fix mechanism had to be considerably improved; new capabilities had to be invented, especially for firmware that had to be exchanged concurrently; more subsystems had to apply unit-concurrent firmware fixes; the executing HMC and SE had to control the CDU tightly.

CDU design decisions

During the early phase of the CDU design, the following basic decisions were made:

- The existing concurrent firmware fix feature allowed—and still allows—the customer to deactivate a firmware fix that was already installed without a planned downtime. The customer had rarely chosen to use the deactivate fix feature. It was decided that CDU would not support the concurrent deactivation of a driver that was already installed. This would significantly reduce the risks for a UIRA.
- Because canceling CDU after it is initiated might result in errors that require a planned disruptive downtime in order to recover with a power-on reset (POR), it was decided that after starting CDU, the new driver must be installed with the CDU process running to completion.
- It was decided that concurrent crossovers from general-availability driver GA^n to driver GA^{n+1} to driver GA^{n+2} must be done sequentially. This means that driver GA^n can be upgraded to driver GA^{n+2} only after driver GA^{n+1} has been fully installed.
- CDU should enable the customer to move concurrently from one specific firmware[2] level—the microcode change level (MCL)[3]—on driver GA^n to a specific level on driver GA^{n+1}.The number of crossover switch points would be limited for each driver pair to reduce the number of transition code versions and to improve test coverage. The next section explains this approach in detail.
- A new firmware driver generally requires more hardware system area (HSA)[4] for the new functions. It was decided that the customer must free used memory if the HSA must be extended. CDU cannot do this automatically. After the driver has been installed, the customer can again use the memory that has not been used for the HSA extension.
- Because there might be some firmware changes that cannot be installed concurrently, such functions must be identified during their initial design so that one of the following actions can be taken: Either significant

[2]Firmware is proprietary code that is usually delivered as microcode. An example of firmware is the basic I/O system (BIOS) in read-only memory on a PC system board. Firmware cannot be modified by the user but can be updated by service personnel.
[3]MCL is a set of changes to firmware. It is functionally equivalent to a software program temporary fix and is intended for broad distribution.
[4]HSA is a logical area of central storage, not addressable by application programs, used to store firmware and control information.

preplanning would have to be performed to ensure that the basics are delivered with the GA1 driver, which is the only driver with a POR, or delivery of the function would have to be held off until the next system generation.

Despite the best preplanning efforts, a problem that remains is that all new functions may not be known during development of the GA1 driver. CDU might require some manual activities, such as turning the configuration of a specific channel on and off or activating or deactivating of a logical partition[5] (LPAR). Disruptive driver upgrades with a planned downtime are permitted at any time. The customer decides which strategy to use to upgrade a system to the next EC driver.

CDU switch point approach

As shown in **Figure 1**, System z9 contains many firmware components. The HMC firmware runs in a separate console and is managed separately from the system firmware. From a CDU perspective, the only requirement for the HMC is to upgrade its firmware from the driver GA^{n} to driver GA^{n+1} prior to beginning the CDU process for the System z9 system firmware.

The System z9 firmware comprises the following main firmware components: support elements (SEs), FSPs, power, LPAR, coupling facility control code (CFCC), i390 code, millicode, and channels. Firmware is a key component to provide flexible solutions and address complex requirements for operations and service. On a System z9, for example, power firmware not only helps manage voltage, it is also executed in the cooling fan hardware. The System z9 firmware components total roughly ten million lines of code. A zSeries firmware driver (i.e., GA^{n}) includes all defined firmware components.

Each firmware component has its own EC stream to release code fixes and new functions to the field. This allows every development team to release code independently, which is important when releasing MCLs. On System z9 GA^{n}, there are 29 unique firmware EC streams.

When CDU is used to upgrade the System z9 firmware, the driver GA^{n} to driver GA^{n+1} will have designated switch points. This means that each GA^{n} firmware EC stream must be at a specified MCL level, and the initial CDU activation can only transition to a specified MCL level for each GA^{n+1} EC stream.

A CDU file called the *min/max file* was invented to define the GA^{n}-from-MCL requirements for each GA^{n} EC stream and the GA^{n+1}-to-MCL requirements for each GA^{n+1} EC stream. The CDU architecture and implementation allow the from-MCL and to-MCL requirements to be a range (i.e., from-MCL 20 to 40), but there was a concern with the amount of testing that would be required to test the different permutations within a range across 29 unique EC streams (for both from and to). Therefore, for each EC stream, a specific switch point defines only one from-MCL level and one to-MCL level.

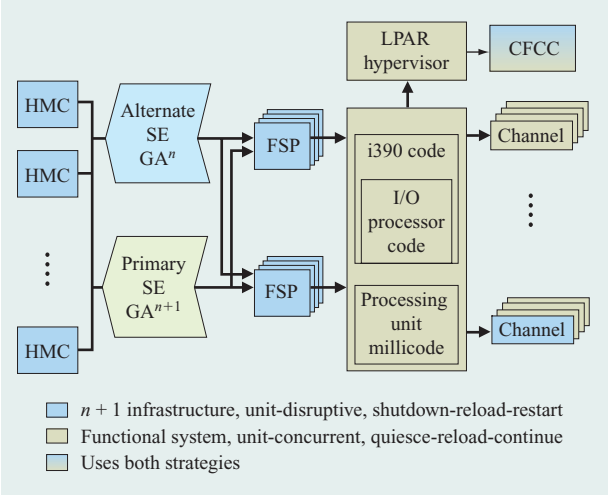

Figure 1

System z9 firmware structure and concurrent patch capability overview.

Additionally, it was concluded that a CDU test could not be performed on every bundle of MCL fixes shipped to the field. Fix bundles are more frequent at the beginning of a GA release compared with the end-of-life time frame of a driver. In order to complete a thorough test across different hardware and operating system configurations, designated CDU switch points have been defined. This is shown in **Figure 2**, in which the boxes represent fix bundle releases and the diamonds indicate the CDU switch points.

Not every fix bundle supports CDU. Hence, as part of the fix-apply process, the operator must make a decision whether or not to apply MCLs above a CDU switch point. The customer must develop a plan that indicates when to use CDU to go to the next GA level and map this plan to an IBM published plan for CDU switch-point release dates. If another scheduled CDU switch-point release is shown to be prior to the customer's targeted GA upgrade date, the customer can apply fixes above the current CDU switch point. However, if no additional CDU switch-point releases are planned prior to the GA upgrade target, the operator should not apply MCLs above the current CDU switch point. IBM does not

[5]A partition in the central electronics complex capable of running its own operating-system image.

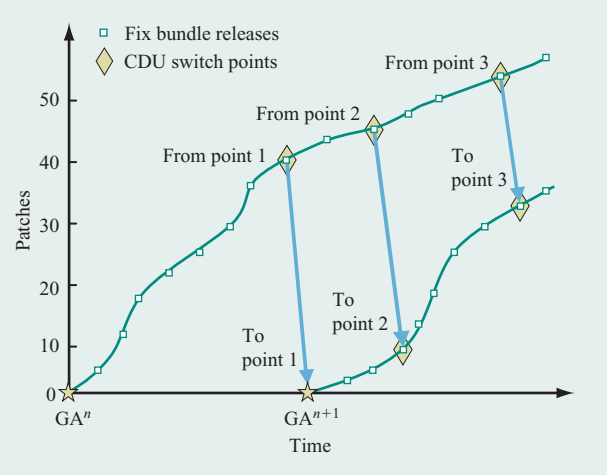

Figure 2

CDU switch points.

recommend removing fixes in order to get back to a CDU switch point.

CDU flow details

A successful CDU consists of the following three steps:

1. *CDU preload step:* This step verifies that the firmware can be upgraded concurrently and installs the correct "from" and "to" firmware levels for the CDU switch points on the primary and alternate SE [3], respectively. The HMC offers several firmware load source options.
2. *CDU activate step:* This step has three phases. The first phase, *preparation*, includes both customer and IBM activities and is performed on the GA^n EC driver. The customer must free the GA^{n+1}-specific amount of memory for additional HSA. The HMC verifies that the CDU is allowed and lets the CDU process gather first failure data capture (FFDC) data, described in more detail in the CDU FFDC approach section below. Finally, the HSA is extended. In the second phase, *transition*, the HMC initiates and controls unit-disruptive and unit-concurrent code switches. New functions are initialized and enabled. The transition phase starts with an alternate and primary SE switch. Some firmware updates require manual intervention. In the third phase, *completion*, the HMC requests such intervention by displaying messages at the end of the activation step. The customer, the IBM product engineer, or both, can perform the requested activity.
3. *Query function availability step:* The status of the added functions is reported. Normally, all new functions are enabled and available.

The following sections describe in detail the activities performed in all CDU phases. The sequence is always controlled by the HMC.

Figure 1 provides an overview of the various subsystems of a System z9 and of how to switch concurrently to a newer firmware version using the unit-disruptive or the unit-concurrent approach. In general, the switching order is left to right, requesting that the subsystems to the left must be able to handle n and $n + 1$ interface traffic.

Preload step

Figure 3(a) shows the main CDU option panel on the HMC. The first step in the CDU process is to initially preload the alternate SE with the GA^{n+1} code while the primary SE and the system continue to operate using the GA^n code. There are two initial preload options: initial preload including MCLs from the IBM Support System, and initial preload only. Both options put the base GA^{n+1} driver on the alternate SE using the CDU DVD-RAM, and the only difference in the two options is that the first option will download any additional fixes that were released after that CDU AROM was released.

The other two preload options allow the download of additional fixes that were not part of the initial preload step. The use of these preload options can enhance CDU performance by avoiding the download of additional fixes over a modem connection, which, for security reasons, is a likely setup in a customer environment.

When the operator selects one of the CDU initial preload options, the HMC validates that the CDU AROM appropriately matches the from-GA^n system EC. The primary SE is requested to validate that the concurrent patch is enabled from previous activities and to retrieve and apply the latest CDU min/max file, ensuring that the latest CDU requirements are known.

Next, a check is made to ensure that the CDU minimum and maximum GA^n requirements are met. If CDU minimum MCL requirements are not satisfied, but no CDU maximum requirement has been exceeded, the operator is given the option to concurrently apply additional GA^n MCLs in order to meet the CDU requirements. If at least one CDU maximum MCL requirement has been exceeded, the operator is told that the CDU is not possible. IBM does not generally recommend removing fixes. The customer must wait for the next CDU switch point.

Once this validation process has completed, the HMC triggers the primary SE configuration and customization upgrade data to be put on the alternate SE upgrade

partition. This ensures that the data is preserved during this transition from GA^{n} to GA^{n+1} code. The alternate SE then downloads from the HMC an image of the operating system partitions and an image of the firmware partitions. Once the download has completed, these two images are unpacked and overwrite the data in all partitions on the alternate SE hard disk except for the upgrade partition. The upgrade data is then restored by taking it from the upgrade partition and transforming it into the GA^{n+1} code structures.

Finally, if the CDU initial preload option included the request to retrieve MCLs, the alternate SE would retrieve from the IBM Support System any additional MCLs that were not part of the CDU AROM. These additional MCLs would be applied on the alternate SE hard disk following certain defined restrictions because the CDU activation process must concurrently manage the firmware updates in memory where that firmware executes. These defined restrictions are derived from certain MCL keywords (such as ACTREQ, which means that a fix must be applied and active before the other fix is applied) as well as from the CDU min/max file. The CDU preload options, which do not include initial preload, can be executed only after the initial preload has been performed.

Activation step

Preparation phase

By selecting the CDU activate option from the main CDU option panel [Figure 3(a)], the following requirements are verified before the code switch starts:

- The concurrent patch feature is enabled.
- The CDU min/max requirements are met (MCLs might have been applied to, or removed from, the GA^{n} code primary SE since the CDU preload).
- No pending conditions from previous CDU or concurrent patch sessions exist.
- There is enough free storage for the additional GA^{n+1} HSA. The driver information contains the CDU GA^{n+1} storage requirements. If there is not enough memory, the customer is told how much memory to free.

After the verification, the HMC needs to know whether the System z9 is ready for the very first or next CDU. Functions that were added by using a previous CDU might require special activities to activate the new code. Such new functions would be available but not yet enabled.

The very first CDU is always allowed. However, the next CDU might require that some new functions that were added with the last CDU be available and enabled. CDU can be performed only on a well-defined

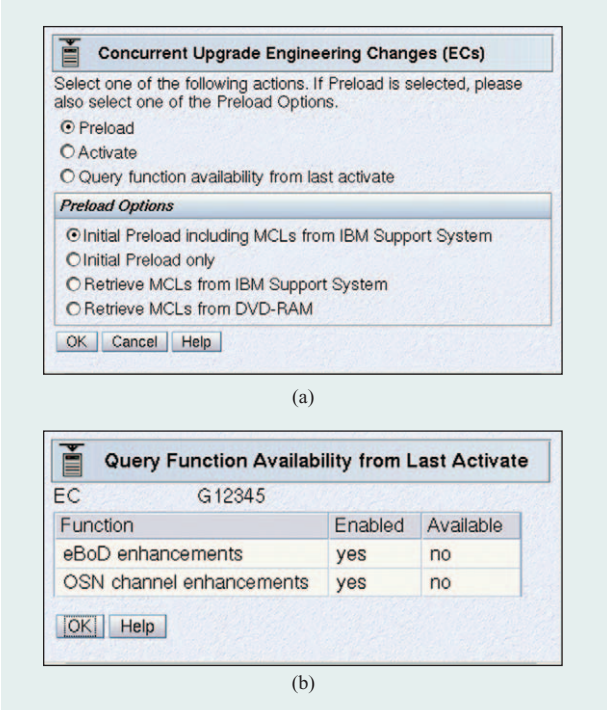

Figure 3

(a) Main CDU option panel; (b) query function availability panel.

"from" firmware level. This implies that the very first step in the CDU preparation phase verifies that the CDU is allowed.

The status of all functions that were added with previous CDU switches is stored in the function availability information (FAI) list. This information is kept over the lifetime of a system. Whether the next CDU is allowed depends on the information in the FAI and is reported to the HMC through a "CDU allowed" indicator.

After verifying that another CDU can be performed, the HMC checks whether the HSA is sufficient for the new functions. The minimum i390 code firmware level contains the information about the amount of memory that is required for each new function. The total amount of memory for the driver switch is calculated by adding the amount of memory required for all new functions. An HMC panel informs the customer about the need for additional free memory if the required HSA exceeds the amount of memory available. The customer must provide memory by deactivating a partition or varying off storage[6] in a partition if the complete memory is used.

[6]Varying off storage is a z/OS* function which allows the operator to free memory. The storage is then free for use by other partitions, controlled by the operator doing a storage vary on.

After verifying the availability of free memory, the HMC issues a command to the i390 code to start the HSA extension. This step marks the point of no return: The firmware and the HSA layout are going to be changed.

The CDU i390 code calls every new function to allocate the required HSA. After each call, the i390 CDU code verifies that there is still enough free memory left to back the HSA requests of all remaining functions that have not yet been called. If there is not enough memory left (most likely caused by a new function requiring more HSA than expected), the HMC stops the CDU activation process and informs the customer about the additional memory required. After the customer has provided the additional memory, the CDU must be started again. The HSA allocation process starts again at the point at which it was stopped.

Transition phase

When the CDU activate preparation phase completes, the transition phase begins. This is the heart of CDU activate because it is in this step that the GA^{n+1} code is applied in each firmware subsystem memory. The SE is the first subsystem to have its GA^{n+1} code loaded, and this is done by executing a CDU alternate SE switch. This causes the GA^{n+1} code to become the new primary SE while the GA^{n} code becomes the new alternate SE. The GA^{n+1} new primary SE must restart and perform a warm-start resynchronization with the other firmware subsystems. This GA^{n+1} SE code must communicate with the GA^{n} code to obtain information about the state of the system. In addition, if service or functional requests occur, the GA^{n+1} primary SE code must be able to interact with the other GA^{n} subsystem code. Once the primary SE completes its warm-start resynchronization, it serially triggers each subsystem to load its GA^{n+1} firmware in the following order: FSP, power, i390 code, millicode, all of the channel subsystems, enhanced initial program loader, LPAR, and CFCC. While additional firmware subsystems transition to the GA^{n+1} firmware, all subsystems must continue to interact with the mixture of GA^{n+1} and GA^{n} code levels. The following is an example of the details of what a firmware subsystem must go through in order to perform the transition from GA^{n} to GA^{n+1} CDU code.

The concurrent patch feature of the i390 code and millicode uses a primary and an alternate code load area to switch between two firmware levels. The executable and linking format (ELF[7]) loader receives a new millicode and a new i390 code load at the start of an i390 concurrent patch sequence. Both code loads are copied into the alternate code load area. The ELF loader and millicode perform some relocation activities to map the new code loads to existing data. Finally, the ELF loader calls millicode to perform the actual code switch.

The existing i390 concurrent firmware patch support of the ELF loader did not allow the addition of new static external variables. However, new functions implemented in i390 depend on the ability to store information permanently and might require that new static variables be created and initialized during the concurrent patch application of i390 code. To reduce the complexity of CDU and avoid any delay during the concurrent patch sequence, the dynamic HSA feature is not used. Instead, the ELF loader allocates spare HSA space at POR time to allow for the growth of i390 code and i390 data area sizes during CDU. The preplanned size of the reserved HSA memory is derived from previous code and data-area changes among drivers.

The ELF loader distinguishes between CDU and the normal i390 concurrent patch case. New function pointers and new remote procedure calls (RPCs) are allowed for CDU, but not for concurrent patches because normal i390 concurrent patches can be deactivated, whereas firmware changes that are applied with CDU cannot be deactivated.

The ELF loader continues with the regular concurrent patch process after the new static variables are allocated and the symbol and relocation tables have been updated accordingly. Function pointers are recalculated using standard relocation methods of the concurrent patch feature. Finally, the ELF loader sets a new event indicator, "CDU i390 switch complete," on all PUs before the switch to the new i390 code load is initiated.

The new i390 code is activated by restarting the i390 environment. The CDU i390 switch complete event is given highest priority, and an ELF loader routine is dispatched that initializes all new static variables. This design ensures that any new i390 code can access the new static data variables only after they have been initialized.

The CDU-relevant ELF loader inventions are described in this paper. For general i390 concurrent patch details, see [4].

Completion phase

Even with a CDU activation success message, additional actions may be required in order to make all CDU functions available or to completely transition all firmware subsystems to GA^{n+1} code levels. The next section presents the details of determining whether all GA^{n+1} functions are enabled and available. Additionally, on System z9 there are currently a few channel subsystems, Crypto Express2 [5], and CFCC that require an additional action to migrate their code from GA^{n} to GA^{n+1}. As part of the CDU activation

[7]ELF is the standard file format used for executable and object files in UNIX**-based operating systems.

process, a hardware message will potentially be displayed directing the operator to invoke two tasks to complete the GA^{n+1} transition for those exception firmware subsystems:

1. The "Query Channel/Crypto Config Off Pending" task allows the operator to see which channels and cryptos must be configured offline and put back online in order to get the GA^{n+1} code loaded. If the System z9 does not have any of the exception channel or crypto hardware installed, no action by the operator is required.
2. The second potential task that the operator may be asked to invoke is the "Query Coupling Facility Reactivations" task. This informs the operator whether any coupling facility (CF) partitions must be reactivated in order to move the CFCC code for that partition to GA^{n+1}.

Query function availability step

Once the CDU activate is complete, the operator can invoke the "Query function availability from last activate" option from the CDU option panel [Figure 3(a)] to get an overview of which functions are not yet available or have not yet been enabled after the CDU completion step. **Figure 3(b)** shows a hypothetical resultant panel display from that invocation. This panel addresses the exception cases in which one or more new GA^{n+1} functions cannot be made available during the CDU activation process.

There are two possible reasons for the occurrence of exception cases. The first is that not all subsystems have transitioned to the GA^{n+1} code level. As described above in the activation step section, the operator should use the "Query Channel/Crypto Config Off Pending" and "Query Coupling Facility Reactivations" tasks to ensure that the GA^{n+1} code-level transitions have completed. Once this GA^{n+1} code-level validation is completed, the operator should invoke the CDU function availability option to see what, if any, functions are still not yet available. The second reason for exceptions is that one or more functions may require some additional action to enable them and make them available. This could range anywhere from having to configure certain types of channels or processors off and on to having to reactivate the system, including a potential update to the input/output configuration data set[8] (IOCDS).

The CDU query function availability panel displays only the functions that are not available. The IBM System z9 Enterprise Class [6] and IBM Resource Link* will describe all new functions that are available as part of the GA^{n+1} release. Additionally, the System z9 overview and IBM Resource Link describe the additional actions required to fully enable and make available any functions that may be displayed on the CDU query function availability panel.

CDU FFDC approach

Existing System z9 FFDC mechanisms are used to save FFDC data so that a CDU-related problem can be debugged without the need for more data. The core of the CDU FFDC approach is a set of three nondisruptive HSA dumps (CECDUMPs) and three nondisruptive LPAR dumps[9]. Each CECDUMP is integrated with the LPAR dump. The first CECDUMP/LPAR dump pair, the pre-HSA-CDU dump, is created before the HSA is changed by CDU. The second dump pair, the post-HSA-CDU dump, is created after the HSA has been extended in preparation for the new firmware driver. The last CECDUMP/LPAR dump pair, the post-CDU dump, is created after the completion of the CDU transition phase.

The first two dump pairs are collected by the original GA^{n} primary SE and transferred to the preloaded alternate SE. The post-CDU CEC dump/LPAR dump pair is collected by the new primary SE, which is already on the GA^{n+1} level. The third CECDUMP/LPAR dump pair is created only if the driver switch is successful. If CDU fails at some point, the nondisruptive LPAR/CEC dump pair is created immediately.

All dumps remain untouched on the system SE until the next CDU is performed, in which case they are replaced with the dumps from the new CDU cycle. The dump files can be manually retrieved as a result by IBM service personnel if an error is reported.

In addition, the SE keeps track of the following events in the system log file:

- POR, CDU start, CDU end, SE reboot.
- Start and end of the various CDU phases.
- Several informational CDU logs with status and flow information.

The system log file, any CDDM[10] dump files, and the security event log file are stored in special directories, together with the CECDUMP/LPAR dump pairs, and remain there until the next CDU cycle is performed.

Benefits of CDU

The major benefit of upgrading firmware by using CDU is that no planned downtime is required. The system can

[8]A configuration definition built by the I/O configuration program and stored on disk files associated with the processor controller.

[9]LPAR dump: Processor Resource/Systems Manager* (PR/SM*) FFDC data collection.

[10]The CEC data debug manager, or CDDM, is the FFDC data manager in the CEC.

continue processing its normal workload without any performance impact.

A typical driver upgrade through CDU, including the SE preload, takes approximately three hours. A normal driver upgrade requires the transfer of the workload to a different system, the shutdown of all applications and operating systems, and a system shutdown. Activating the system again with the new firmware driver, including software IPL, might then take up to eight hours. By using CDU, the customer can not only continue using the system while it is being upgraded, but also save significant time compared with a traditional driver upgrade.

Conclusion and future direction

With the introduction of the CDU feature, IBM System z9 customers have the ability both to upgrade zSeries firmware and to add new functions without the need for planned downtime. This paper described CDU switch points, which are defined to identify the point at which the transition to a new GA driver level can occur. The CDU process itself consists of three underlying steps: preloading the CDU switch-point firmware levels on the primary and alternate SEs, activation of the new driver level, and querying function availability to determine whether all new functions are ready for use.

Improvements are planned for CDU with each new System z generation. However, there will be no major firmware redesign as a result of our goal to provide active support. Subsequent System z systems will provide incremental enhancements such as improved concurrent firmware patch control and more subsystems with true concurrent patch capability. Some I/O adapters without true concurrent patch capability will no longer be part of the system offering but will be replaced by modernized versions with true concurrent patch support.

The addition of CDU technology contributes to the continuous reliable operation rate of the IBM System z9 and is another step forward in reducing downtime.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of The Open Group in the United States, other countries, or both.

References

1. B. E. Casey, G. L. Dunlap, M. C. Enichen, D. A. Larnerd, J. A. Morrell, S. R. Nichols, P. D. Pagerey, and S. L. Rockwell, "A Method and System for Providing a Common Hardware System Console Interface in Data Processing Systems," U.S. Patent 6,182,106, January 30, 2001.
2. J. M. Nick, B. B. Moore, J.-Y. Chung, and N. S. Bowen, "S/390* Cluster Technology: Parallel Sysplex*," *IBM Syst. J.* **36**, No. 2, 172–201 (1997).
3. B. D. Valentine, H. Weber, and J. D. Eggleston, "The Alternate Support Element, a High-Availability Service Console for the IBM eServer* z900," *IBM J. Res. & Dev.* **46**, No. 4/5, 559–566 (2002).
4. C. Axnix, T. Hendel, M. Mueller, A. Nuñez Mencias, H. Penner, and S. Usenbinz, "Open-Standard Development Environment for IBM System z9 Host Firmware," *IBM J. Res. & Dev.* **51**, No. 1/2, 195–205 (2007, this issue).
5. T. W. Arnold, A. Dames, M. D. Hocker, M. D. Marik, N. A. Pellicciotti, and K. Werner, "Cryptographic System Enhancements for the IBM System z9," *IBM J. Res. & Dev.* **51**, No. 1/2, 87–102 (2007, this issue).
6. IBM Corporation, "IBM System z9 Enterprise Class"; see *http://www-03.ibm.com/systems/z/z9ec/*.

Received March 10, 2006; accepted for publication June 15, 2006; Internet publication December 5, 2006

Andreas Muehlbach *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (muhlbach@de.ibm.com)*. Mr. Muehlbach holds a Dipl.-Ing. degree in electrical engineering and computer science from the Technical University of Kaiserslautern. He is a Senior Development Engineer working in processor firmware development for IBM 9221, CMOS G1 to G6 processors, z900, z990, and z9.

Brian D. Valentine *IBM Systems and Technology Group, 1701 North Street, Endicott, New York 13760 (bdvalent@us.ibm.com)*. Mr. Valentine graduated from Pennsylvania State University with a B.S degree in computer science. He is a Senior Technical Staff Member Programmer working in HMC and SE Licensed Internal Code development.

Daniela Immel *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (immel1@de.ibm.com)*. Mrs. Immel received her B.S. degree in business information systems from the University of Applied Sciences, Karlsruhe. She is a Staff Development Engineer in zSeries processor firmware.

Michael S. Bomar *IBM Systems and Technology Group, 1701 North Street, Endicott, New York 13760 (bomarms@us.ibm.com)*. Mr. Bomar received a B.S. degree in computer science from the University of Tennessee. He is an Advisory Programmer working in HMC and SE Licensed Internal Code development.

Timothy V. Bolan *IBM Systems and Technology Group, 1701 North Street, Endicott, New York 13760 (bolant@us.ibm.com)*. Mr. Bolan graduated from Rensselaer Polytechnic Institute with a B.S. degree in electrical engineering. He is a Senior Engineer working in HMC and SE Licensed Internal Code development.