

百硕同兴

Bayshore Advisor

客
户

通
讯

冬

总第 26 期（2011年12月1日）

百硕同兴科技（北京）有限公司

本期特写

——2011 百硕同兴主机性能调优技术研讨会

随着国内主机用户金融业务的迅速发展，对 IT 技术的依赖和系统管理水平的要求越来越高。百硕同兴从客户角度出发，历时大半年筹备，今年再次邀请到了合作伙伴---EPS 公司(Enterprise Performance Strategies, Inc.) 总裁、美国资深主机技术专家 Peter Enrico 来中国，于 11 月中旬举办了两场主题为 Parallel Sysplex Performance Tuning 的高级技术研讨会（2011 年 11 月 14—15 日在上海、17—18 日在北京），主要内容涵盖了 Parallel Sysplex、CF、Data Sharing 方面的性能分析与调优。共有 80 余位客户参加了此次研讨会。

研讨会主讲人 Peter Enrico 先生，在主机高端技术领域具备丰富的理论基础和实践经验，并擅长技术专题研讨与培训。从整体意见反馈来看，客户普遍认为内容非常有帮助，并希望能有更多类似的高水平专家讲座、培训及技术交流。

长期以来，百硕同兴在致力于提供高品质服务的同时，不断加大投入提高员工的专业水准，也在努力尝试与国内客户分享自身的成功经验和业界的最新技术。今后，我们将继续努力，持续为国内主机用户创造更多高水平的技术交流与培训机会。



■ 上海及北京的研讨会现场

本期主要内容

消息快递 *News Express*

SMF 113 Records and Some New Indicators for z/OS 4
百硕外籍技术专家 Martha Hall

经验分享 *Experience & Tips*

如何跨地址空间访问 CICS 的 Control Block 8
百硕工程师 尹支玉

浅谈 RMM 中 VRS POLICY 的测试方法 16
百硕高级工程师 马彤雷

百硕博客 *Bayshore Blog*

打造自己的 CPU 使用率监控工具 23
百硕高级工程师 陈银波

清理 GLOBAL TSQ 的几种方法 31
百硕工程师 陆书博、王军波

OMEGAMON For MVS Command 使用简介 34
百硕工程师 许扬

专家问答 *Q & A with Experts*

数据传输中常见问题的解答 41
百硕工程师 王晨

SMF 113 Records and Some New Indicators for z/OS



■ 文 / 百硕外籍技术专家 Martha Hall

z10 and z196 and other new processors have introduced an architectural structure which is significantly different from earlier systems. Multiple CPU books and their caching functions have added a new level of complexity to performance analysis and capacity planning. IBM has introduced some new indicators to examine in this environment.

We are going to evaluate 4 of these new metrics in this document.

This data is recorded in the SMF113 record called the "Hardware Capacity, Reporting and Statistics" record. The hardware data event counters are recorded in the SMF113 subtype 2 record. For each hardware data event collection cycle, the system creates one subtype 2 for each active CPU. The system captures the valid counters and places them contiguously in subtype 2 of record type 113. There are four groups of counters contained in the SMF113. These are basic counters, problem counters, crypto counters, and extended counters.

You can find the SMF113 record layout in the "MVS System Management Facilities (SMF)".

You can use the SMF113 data to compare performance before and after hardware and software upgrades, WLM changes, LPAR configuration changes, and applications changes to measure the impact of each change. The SMF113 records should be used and evaluated during any critical system test.

Relative Nest Intensity (RNI)

RNI is a new capacity planning metric introduced by IBM to be used to determine the workload type when using the z/PCR Capacity Planning tool. z/PCR can use the SMF113 records to establish the Relative Nest Intensity and use that ratio to determine the workload MIPs for a z10/z196 LPAR. These ratios are published by IBM at the LSPR website. The URL is <https://www-304.ibm.com/servers/resourceink/lib03060.nsf/pages/lsprindex?OpenDocument>.

Capacity performance has always been more closely associated with how a workload uses and interacts with a particular processor hardware design. With the availability of CPU MF (SMF 113) data on z10 and z196, you can determine the interaction of workload and hardware design in your production workloads.

There are three new workload capacity categories. These categories are based on memory hierarchy. The instructions and the data for programs to be executed on a processor are held in a HSB (High Speed Buffer) which can also be referred to as cache. Some of these caches are limited to one processor and some are shared between several processors. A memory "nest" for a

System z processor refers to the shared caches and memory along with the data buses that connect the shared cache.

In this environment, the best performance is achieved when the data and instructions are located in the cache that is nearest to the processor. If you are moving workloads from one processor to another, the performance will vary as the memory hierarchy changes from one processor to another. Other aspects that can have an impact on performance at the same time are locality of reference, IO rate, contention from other LPARs and applications.

The most performance sensitive area of the memory hierarchy is the activity to the memory nest, which is the distribution of activity to the processors shared caches and memory. The term “Relative Nest Intensity (RNI)” is used to indicate the level of activity to the memory hierarchy.

You can use the data from CPU MF to establish the RNI of the workload running in an LPAR. If an RNI is high, the deeper into the memory hierarchy the processor must go to retrieve the instructions and data for that workload. Optimum performance is achieved when the data and instructions are located near to the cache. If this is true then the RNI for an LPAR and its workload is low.

This diagram shows the workload categories and the expected RNI.

LOW RNI	Workload Category Contributors	HIGH RNI
Batch	Application Type	Transactional
Low	IO Rate	High
Single	Application Mix	Many
Intensive	CPU Usage	Light
High Locality	Data Reference Pattern	Diverse
Simple	LPAR Configuration	Complex
Extensive	Software Configuration Tuning	Limited

Data reference pattern and CPU usage are determined primarily by the application. LPAR configuration will also determine the RNI for any particular system. Having more LPARs can negatively impact the RNI.

The Software Configuration category can be adjusted to influence the RNI. The number of address spaces that are needed to support a workload can raise a workload’s RNI as the working set of instructions and data from each address space increases the competition for the processor caches. Tuning to reduce the number of simultaneously active address spaces to the proper number needed to support a workload can reduce RNI and improve performance. In this case, less is better.

As mentioned earlier, z/PCR and other vendor programs can be used to calculate the RNI from the CPMF (Central Processing Measurement Facility) data.

The calculation for a z196 uses the fields in CPMF to ascertain the RNI.

For z196 the CPU MF factors needed are:

- L3P: percentage of L1 misses sourced from the shared chip-level L3 cache
- L4LP: percentage of L1 misses sourced from the local book L4 cache

- L4RP” percentage of L1 misses sourced from a remote book L4 cache
- MEMP: percentage of L1 misses sourced from memory

The formula for the z196 is:

$$\text{z196 RNI} = 1.6 \times (0.4 \times \text{L3P} + 1.0 \times \text{L4LP} + 2.4 \times \text{L4RP} + 7.5 \times \text{MEMP}) / 100$$

The following table illuminates the three new workload categories.

Workload Category	Workload Category Description
LOW RNI	A workload category representing light use of the memory hierarchy.
AVERAGE RNI	A workload category representing average use of the memory hierarchy. This would be similar to the past LoIO-mix workload and is expected to represent the majority of production workloads.
HIGH RNI	A workload category representing heavy use of the memory hierarchy.

The RNI can be calculated by using z/PCR. There are programs available from other vendors to calculate RNI such as MXG.

CPI (Cycles Per Instruction)

This number can be obtained from the SMF113 record.

It is recommended to look at the cycles per instruction after any sort of hardware or software upgrade. You want to see this metric decrease as changes occur. If CPI increases it means that it is taking more processor cycles to execute instructions in your environment. The fewer cycles are used, the better the performance.

It is normal to look at this metric before and after Hiperdispatch is implemented to measure the delta. The formula for CPI is cycle count divided by instruction count. Data counters from the SMF113 subtype 2 record which are needed for the metric are as follows:

BC0 provides the cycle count. BC1 provides the instruction count.

$$\text{CPI} = (\text{Cycles} / \text{Instructions})$$

LPAR Physical Busy

This formula uses the CPU speed in cycles per microsecond value from SMF113_2_CPSP. This value is recorded for **each** CPU.

$$\text{LPARCPU} = ((1/\text{CPSP}/1,000,000) * \text{B0}) / \text{Interval Seconds} * 100$$

Example Calculation where:

Processor Speed (cycles per microsecond) = SMF113_2_CPSP is 4404

BC0: Cycle Count is 2917645110273

Interval seconds is 900 (15 minute interval)

Then the percentage of CPU can be derived from the 113 record as follows:

$$(((1/4404/1000000)*2917645110273)/900)*100 =73.61.$$

This number can be expressed as a percentage of one CPU by adding the fields for each CPU or it can be expressed as an average for all CPUs by dividing the total for all CPUs by the number of CPUs.

Ratio of Problem State to Supervisor State (The Stability Index)


This measurement is useful to compare before and after application changes. I think it is as important as measuring the capture ratio during application changes.

SMF113 data required is the field that contains the total of supervisor state instructions and the problem state instructions. This value is contained in the BC1 counter. The number of problem state instructions is contained in the PC33 counter.

The ratio is obtained by dividing the value of the PC33 counter by the BC1 counter.

This is useful in comparing performance before and after an application change. We want to ensure that the ratio obtained is reasonably stable. We want to show that the application has not significantly impacted the way that it interacts with the operating system functions.

To elaborate, consider if an application changes the way IO is managed in such a way that the number of IOs is increased. In this case the ratio would decrease because of the additional system state instructions required to manage the IO.

There are many other useful indicators contained in the SMF113 records. These indicators are important since they address the new architectural functions with the newer hardware. 

如何跨地址空间访问 CICS 的 Control Block



■ 文 / 百硕工程师 尹支玉

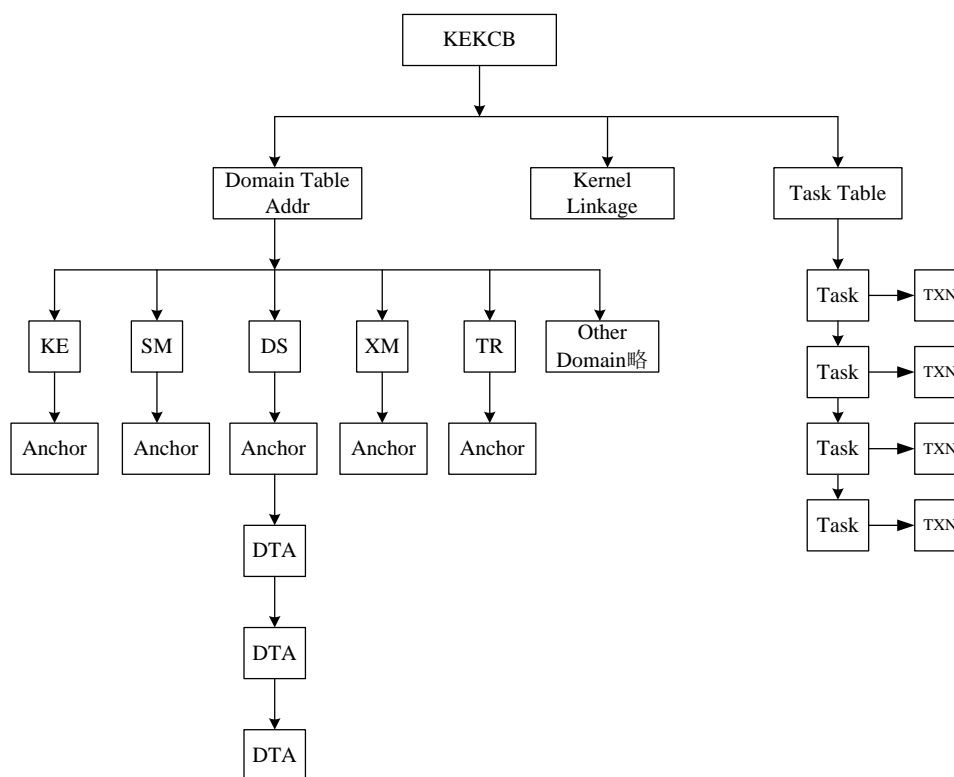
一、概述

Control Block 是 CICS TS 内部维护的数据结构，包含了 CICS 运行中的重要信息，了解 CICS 的 Control Block 及其关系对于我们理解 CICS 的运行机制、问题分析诊断都有很好的帮助。基于安全及系统稳定性的考虑，CICS 本身不提供用户直接对 Control Block 进行访问，但我们可以通过 CICS System DUMP 获取某一时刻的内存数据来了解 CICS 的 Control Block，也能从其它地址空间（如 TSO 或 Batch）进入 CICS Region 地址空间直接访问其 Control Block。本文通过实例介绍了 CICS 主要 Control Block 间的关系以及利用 MVS 的跨地址空间功能访问它们的方法。

二、CICS TS 的主要 Control Block 及其关系

CICS TS 按照不同的功能划分为不同的 Domain，每一个 Domain 负责相对单一的功能，如 KE Domain 为 CICS 的核心（Kernel）部件，所有执行的任务都需要在 Kernel 中注册，其它 Domain 之间的通讯也需要向 Kernel 请求服务，DS Domain 负责任务调度，XM domain 负责交易管理，SM domain 负责内存管理，LG Domain 负责读写 CICS log 等等。对于运行中的 CICS 来说，Domain 包括一组执行代码和与之有关的内存结构，即 Control Block。作为一个复杂的交易处理环境，CICS 的 Domain 必然要与之相关的其它 Domain 通讯以传递必需的信息。在设计上，每一个 Domain 不能直接操作其它 Domain 的 Control Block，只能通过 Kernel Linkage 调用属于其它 Domain 的 Gate（Domain Interface）请求相关的服务。

Control Block 包含了 CICS 运行时的重要信息，CICS 核心通过 Control Block 将各个 Domain 关联起来形成一个有机的交易处理环境。下面是 Control Block 之间的逻辑关系图：



KEKCB 是所有 Control Block 的 Anchor，它包括 Domain Table、Kernel program 以及 Task Table。Task Table 和 Domain Table 由 Kernel Domain 分配、维护，所有 Task Table 形成一个链表，用不同的 Task Table Number 标识。Domain Table 中包含指向相应 Domain Anchor 的地址，可以通过 KCB 找到 CICS Domain 的 Anchor，进一步获得该 Domain 的 Control Block，然后从相关的 Control Block 中得到每一个 Domain 的详细信息，如从 DSANCHOR 得到由 Dispatcher 管理的 Task 信息；从 XMANCHOR 得到 Transaction Manager 管理的交易信息；从 SMANCHOR 中得到 Storage Manager 管理的 CICS 内存信息。实际上，CICS 核心也正是通过这些 Control Block 之间功能上的关联为我们建立起一个完整的交易处理环境。

这些 Control Block 由 CICS 内部使用，用户不能对 Control Block 直接访问，且这种保护机制变得愈加严格。如从 TS 3.1 开始，就不允许用户程序直接访问 CSA（之前的版本可以通过 TCA 中的 TCACSAAD 找到 CSA，现在该数据项已经变为 Fetch Protected），有时通过使用 DFHKERN 宏可以访问 CSA，但 IBM 不提供对其使用的用户支持。通过 CICS 提供的一些 API 或 SPI 来实现一些功能，但这些编程接口都不足以获得 Control Block 中的足够信息。

我们知道 z/OS 中所有执行的任务都有自己的地址空间，操作系统在任务存续期间维护地址空间的信息，可以通过 CVT→ASVT→ASCB 找到 CICS 的地址空间。作业提交后，z/OS 为其创建地址空间，将作业的 JOBNAME、ASID 等重要信息记录到 ASCB 中。因此我们可以从 ASCB 中找到 CICS 的 ASID，通过 ASID 从 CICS 之外的地址空间进入 CICS 内部访问到其中的 Control Block。

三、跨地址空间访问 Control Block

以下用一个简单汇编程序读取 Kernel Task 链中的数据项，得到当前系统中已经建立的 Task 的相关信息，以初步了解 CICS Control Block 间的内部联系。

1. 通过 CVT 取得 CICS Region 的 ASID

```

.....
SAVE (14,12)
USING CICSCB,R12
LR R12,R15
LR R14,R13
LA R13,SAVEAREA
ST R13,8,(R14)
ST R14,4,(R13)
LR R5,R1
L R6,0(R5)
MVC CICSJOB(8),2(R6)
L R5,16 (1)
L R5,CVTASVT-CVTMAP(R5) (2)
USING ASVT,R5
LA R6,ASVTENTY (3)
L R8,ASVTMAXU (4)

```

指令 (1) 从 PSA 中 LOAD CVT

(2) 从 CVT 中得到 ASVT 地址

(3) 第一个 Address Space 的 ASID 地址

(4) 系统中当前设置的最大的 Address Space 个数

```

TM 0(R6),ASVTAVAL (5)
BO GET_NEXT_ITEM
L R9,0(R6)
USING ASCB,R9
L R1,ASCBJBNS *start task jobname (6)
LTR R1,R1 (7)
BZ GET_NEXT_ITEM (8)
CLC CICSJOB(8),0(R1) (9)
BE MATCHED_CICS (10)

```

指令 (5) 操作数常量 ASVTAVAL 的值为 X'80'，TM 指令判断该 Entry 是否已被已建立的地址空间占用，如果该 Entry 为空则检查下一个 Entry 是否为 CICS。

指令 (6) ~ (10) 用来判断当前找到的地址空间是否为所请求 CICS Region。一般来说，在实际的客户化中 CICS 都是以 Started Task 启动，ASCBJBNS 存放的是 Started Task 的 Jobname，因此只需要判断 ASCBJBNS 即可。通过循环执行这段指令序列来匹配 CICS 的地址空间，得到对应的 ASID。

2. 根据上面获取的 CICS ASID 进入 CICS 内部

由于发生跨地址空间（Cross Memory Access）的行为，我们要以 CPU 特权指令模式进入 CICS 的地址空间执行 SAC 指令访问 Secondary Address Space:

```

.....
MVC ASID,ASCBASID (11)
MODESET MODE=SUP,KEY=ZERO (12)
AXSET AX=IDX1 (13)
LH R4,ASID (14)

```

SSAR R4	(15)
LAM R10,R10,=F'1'	(16)
SAC 512	(17)
.....	

指令（11）~（16）是进行进入 AR MODE 访问 Secondary Address Space 的固定模式，ASCBASID 的值为 CICS 的 Address Space Number。

（12）设置 CPU 的特权模式

（13）IDX1 取值为 1，表示 Secondary Address Space，我们将从 Home Address Space 切换到 Secondary Address Space。

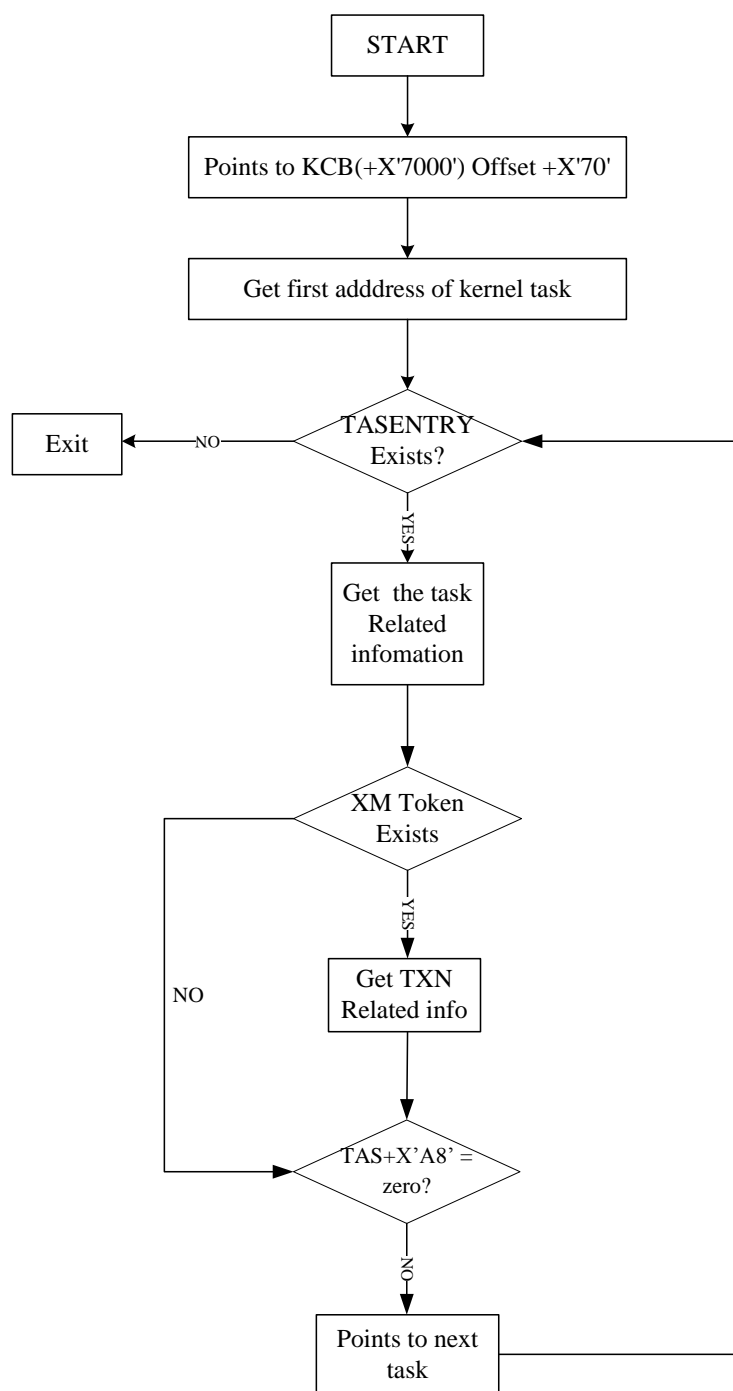
（15）R4 中为 CICS 的 ASID，调用 SSAR 设置其为 Secondary Address Space

（16）LOAD ALET='00000001' 到 Access Register 10 中

（17）切换到 AR MODE，接下来通过 R10 访问 CICS 中的 Control Block

3. 从 Control Block TASENTRY 中获取 Task 信息

进入 CICS 的地址空间后，可以通过 Kernel Control Block Anchor (KCB) 逐级访问获得当前系统中所有 Task 的信息。CICS TS 的 KCB 地址为 X'00007000'。KCB 偏移 X '78' 处为 Domain Table Header，其中包含 Domain Table 的起始地址和结束地址，X '70' 处为已经被 Kernel 创建的任务的起始地址，指向 TASENTRY。Control Block TASENTRY 中包含的 Task 信息有 TCA Address、Task State、KTCB Address、Task CPU Time、Task Runaway Left Time 等，偏移 X'A0'处为 XMTXN 的地址，偏移 X'A8'处为 Task 链中的下一个 Task 地址，获取这些数据的流程如下：



选取 TASENTRY 中部分数据项的偏移:

Tasentry + X'28' : TCA Address

+X'3C' : Task State

+ X'40' : KTCB address

+ X'90' : CPU time of this task

+ X'98' : the runaway left time if set RUNAWAY value(SIT /transaction)

+ X'A0' : XM transaction token address

+ X'A8' : next TASKENTRY address

+ X'C0' : the TCB ID used by this task

选取 XMTXN 中部分数据的偏移:

+ X'3C' : Task Number

+ X'48' : Tran ID

对应的主要汇编指令:

```

MAIN_PROC MVC KEKCB,=XL4'00007000' (18)
        L R10,KEKCB
        A R10,=F'112' (19)
        L R10,0(R10)
        DROP R9
        XR R9,R9
        GETMAIN RU,LENGTH=640000,LOC=(31,31)
        LR R7,R1
        ST R1,TMPAREA
GET_TASK_INFO DS 0H
        CLC 0(8,R10),=CL8'TASENTRY'
        BNE NO_MORE_TAS
        A R9,=F'1'
*****
*以下为从 Control Block TASENTRY 获取的信息 *
*****
        MVC 0(4,R7),X'28'(R10) TCA ADDRESS (20)
        MVC 4(1,R7),X'3C'(R10) TASK STATE
        MVC 5(4,R7),X'40'(R10) KTCB ADDRESS
        MVC 9(8,R7),X'90'(R10) CPU TIME USED(CLOCK)
        MVC 17(2,R7),X'98'(R10) RUNAWAY LEFT
        MVC 19(4,R7),X'A0'(R10) XM TXN TOKEN ADDRESS
        MVC 23(2,R7),X'C0'(R10) TCB MODE NAME (21)
*****
*以下为从 Control Block XMTXN 获取的信息 *
*****
        L R8,X'A0'(R10) (22)
        LTR R8,R8
        BZ TO_NEXT_TAS
        LR R11,R10
        L R10,X'A0'(R10)
        MVC 25(4,R7),X'3C'(R10) TASK NUMBER
        MVC 29(4,R7),X'48'(R10) TRAN ID
        MVC 33(8,R7),X'50'(R10) XM ATTACH TIME
        MVC 41(8,R7),X'58'(R10) TIME WAITED FOR TCLASS
        MVC 49(8,R7),X'60'(R10) TIME WAITED FOR MXT
        LR R10,R11 (23)
TO_NEXT_TAS DS 0H
        LA R7,57(R7)
        L R10,X'A8'(R10) NEXT TASENTRY
        LTR R10,R10 LENGTH = 57
        BNZ GET_TASK_INFO
NO_MORE_TAS DS 0H
        EPAR R2 (24)
    
```

```

SSAR R2 (25)
SAC 0 (26)
AXSET AX=IDX0 (27)
MODESET MODE=PROB,KEY=NZERO (28)
*****
* 输出获取的数据 *
*****
L R7,TMPAREA (29)
OPEN (SYSPRINT,OUTPUT)
PRINT_TASK_INFO DS 0H
MVC TASK_INFO,=CL80' '
UNPK TASKNUM,25(5,R7)
MVC TASK_INFO,TASKNUM TASK NUMBER
MVC TASK_INFO+8(8), R8,0(R7) TCA ADDRESS
MVC TASK_INFO+17(4),29(R7) TRAN ID
MVC TASK_INFO+22(2),23(R7) TCB MODE NAME
MVC TASK_INFO+25(8), R8,5(R7) KTCB ADDRESS
.....
MVC TASK_INFO+34(19),ATCHTME TASK ATTACH TIME
MVC TASK_INFO+54(10),WTCLTME TIME WAITED TCLASS
MVC TASK_INFO+65(10),WMXTTME TIME WAITED MXT
L R5,5(R7)
LTR R5,R5
BZ NEW_LINE
PUT SYSOUT,TASK_INFO
NEW_LINE DS 0H
LA R7,57(R7)
LTR R7,R7
BZ END_LINE
BCT R9,PRINT_TASK_INFO
END_LINE DS 0H
L R7,TMPAREA
.....

```

指令 (18) CICS TS 的 KEKCB 的地址 (CICS TS 为偏移 X'7000')

(19) 偏移 X '70' 处指向 TASENTRY Control Block，其中包含已经被 KE 建立的任务的信息，偏移 X'A0'处为 Control Block XMTXN 的地址，其中包含与 Transaction 有关的信息。

(20) ~ (21) 从 Control Block TASENTRY 中获取数据

(22) ~ (23) 从 Control Block XMTXN 中获取数据

(24) ~ (28) 将控制切回 Primary Address Space 并退出 AR MODE

4. 程序运行环境要求及注意事项

由于执行 MODESET Macro、AXSET、SSAR、SAC 指令需要 CPU 运行在 Supervisor 状态，因此 Link Edit 时要加参数 AC=1，并将 LOAD MODULE 放到 SYS1.LINKLIB 或其他系统 APF 授权库中，然后通过提交 Batch Job 方式执行。

本文中涉及到的 CICS Control Block 的结构、地址偏移量参考的是 CICS TS 3.2，其它版本的 CICS 可能会有所差异。B

浅谈RMM中 VRS POLICY的测试方法



■ 文 / 百硕高级工程师 马彤雷

DFSMSrmm（以下简称 RMM）是 IBM z/OS 中 DFSMS 的重要组成部分，设计之初是希望通过 OAM 等相关系统软件实现 IBM 3494 等自动磁带库的管理。目前越来越多的客户将 RMM 嵌入到 EMC 虚拟磁带库或 STK 磁带库的整体架构中，通过 RMM 实现对备份数据集生命周期的自动化管理。

VRS（Vital Record Specifications）作为 RMM 中管理数据集生命周期的重要技术手段，越来越多的被广大存储管理员所采用。本文将结合笔者在日常使用过程中的经验，与大家共同探讨如何通过更加有效的测试方法，直观地了解 VRS POLICY 与备份数据集之间的相互关系。

一、新增 VRS POLICY 测试

1. 当前系统中存在如下数据集

Data set name	Volume Serial	Assigned date	Expiration date	Status
IBMUSER. CYC. C01V0001	000008	2011/300	2011/320	MASTER
IBMUSER. CYC. C01V0002	000009	2010/301	2011/320	MASTER
IBMUSER. C01V0001	000006	2011/140	2011/320	MASTER
IBMUSER. C01V0002	000007	2011/340	2011/320	MASTER
IBMUSER. DAYS. EXP	000000	2011/295	2011/320	MASTER
IBMUSER. DAYS. NOEXP	000001	2011/311	2011/320	VRS
IBMUSER. GDG. G0001V00	000004	2011/299	2011/320	MASTER
IBMUSER. IGNORE. EXPDT	000014	2011/315	2011/329	MASTER
IBMUSER. LREF. EXP	000002	2011/205	2011/320	MASTER
IBMUSER. LREF. NOEXP	000003	2011/205	2011/320	MASTER
IBMUSER. SPEC. EXP. EXAMPLE	000010	2011/279	2011/320	MASTER
IBMUSER. SPEC. NOEXP. EXAMPLE	000011	2011/279	2011/320	MASTER
当前系统日期：2011. 315				

2. 当前系统中只有一个 VRS POLICY

从以下定义可以看出，IBMUSER.DAYS.**是一个 DSN VRS，满足这条 VRS POLICY 的数据集通过 VRS 被保留 5 天，且 VRS POLICY 定义将被系统永久保留。

Data set mask . : 'IBMUSER.DAYS.**'	GDG . : NO
Job name mask . :	
Count . . . : 5	Retention type : DAYS
	While cataloged : NO
Delay . . . : 0 Days	Until expired : NO

```

Location .....: HOME
Number in location : 5
Priority .....: 0

                                Release options:
Next VRS in chain .:   Expiry date ignore ...: NO
  Chain using ...:    Scratch immediate .....: NO

Owner .....: IBMUSER
Description ..:
Delete date ..: 1999/365 (YYYY/DDD)

```

3. 当前数据分析

DSN 类型的 VRS POLICY 通过 DATA SET MASK 来匹配备份数据集。当前系统中 IBMUSER.DAYS.EXP 和 IBMUSER.DAYS.NOEXP 两个数据集的名称都符合 IBMUSER.DAYS.** 的命名规则，都应该由这条 VRS 所保护。但从数据集所对应的 VTV 状态看，只有 IBMUSER.DAYS.NOEXP 所对应的 000001 的 STATUS 为 VRS，即该数据集当前正由 VRS POLICY 保护，而 IBMUSER.DAYS.EXP 所对应的 000000 的 STATUS 为 MASTER，不受 VRS 保护但也没有被 SCRATCH。

这究竟是因为什么？难道是 RMM 出错了吗？其实原因很简单，RMM 中保护数据集的方法主要可以分为 VOLUME 到期日和 VRS 两种，而这个数据集是 2011/295 创建的，今天是 2011/315，IBMUSER.DAYS.** 这条 VRS 设置保护数据集的时间是 5 天，也就是说该 VRS 已经不再保护这个数据集了。但是该 VRS 中 Expiry date ignore 这个参数的设置为 NO，这就意味着任何对应此 VRS 保护的数据集，在失去 VRS 保护后，它的到期日将由 VTV 的过期日来决定，而该数据集所对应的 VOLUME 的到期日是 2011/320，所以状态仍然为 MASTER。

4. 新增 VRS

```

Data set mask .: 'IBMUSER.SPEC.**'          GDG .: NO
Job name mask .:

Count ....: 50          Retention type .....: DAYS
                                While cataloged .....: NO
Delay ....: 0 Days      Until expired .....: NO

Location .....: HOME
Number in location : 5
Priority .....: 0

                                Release options:
Next VRS in chain .:   Expiry date ignore ...: NO
  Chain using ...:    Scratch immediate .....: NO

Owner .....: IBMUSER
Description ..:
Delete date ..: 1999/365 (YYYY/DDD)

```

为了便于理解，以上仍以最简单的 DSN VRS 为例，这条 VRS POLICY 会将符合 IBMUSER.SPEC.** 命名规范的数据集保留 50 天，然后再依据 VTV 的到期日决定是否继续保留此数据集。

5. 新增或修改 VRS 后的测试

一般在新增 VRS 后，都不会直接执行 ‘VRSEL’ 的动作，因为有可能新增或修改的 VRS POLICY 对数据集的影响并不如你所愿，所以在新增 VRS POLICY 后，一般都会首先执行 ‘VERIFY’，或者出于安全考虑，指定 RMM 的系统参数 VRSCHANGE 为 VERIFY，强制系统中 VRS 发生变化后必须先执行 ‘VERIFY’。

接下来我们看看通过 ‘VERIFY’ 到底可以看到些什么？

5.1 执行以下 RMM HSKP 作业：

```
//EDGHSKP1 EXEC PGM=EDGHSKP,
//    PARM='VERIFY'
//SYSPRINT DD SYSOUT=*
//MESSAGE DD DSN=RMM.MESSAGE.OUTPUT,DISP=SHR
//ACTIVITY DD DSN=RMM.ACTIVITY.OUTPUT,DISP=SHR
//REPORT DD DSN=RMM.REPORT.OUTPUT,DISP=SHR
```

5.2 测试结果初步分析

在作业成功结束后，我们先来看看 REPORT 输出。浏览 REPORT DD 所对应的数据集。在如下报告中我们可以清晰的看到，当前 RMM 中所有受 VRS 保护的数据集名称、保存日期、对应 VRS POLICY 的名称和定义等，当然也包含了我们新增加的 VRS POLICY，以及它所影响到的数据集名称。

REMOVABLE MEDIA MANAGER		VITAL RECORDS RETENTION REPORT										PAGE 1	
Copyright IBM Corp. 1993, 2007		-----										TIME 08:41:25 DATE 11/11/2011	
JOB MASK	DATA SET OR VOLUME MASK	OWNER	TYPE	RETN	C	X	DELETE	DLY	COUNT	STNUM	LOCATION	RLSE	LASTREF
CONT:- _													
	IBMUSER.DAYS.**	IBMUSER	DSN	DAYS	N	N	31/12/1999	0	5	5	HOME		
CONT:- 11/11/2011													
JOB NAME	DATA SET NAME	2ndVRS	2ndNAME	FSEQ	DSEQ	VOLUME	VSEQ	OWNER	CURRENT	REQUIRED	PRTY	RETDTE	RETNAME
	IBMUSER.DAYS.NOEXP			1	0	000001	1	IBMUSER	SHELF	SHELF	5000	12/11/2011	*
CONT:-													
NUMBER OF DATA SETS RETAINED (GROUP STORE) =										1	0		
JOB MASK	DATA SET OR VOLUME MASK	OWNER	TYPE	RETN	C	X	DELETE	DLY	COUNT	STNUM	LOCATION	RLSE	LASTREF
CONT:- _													
	IBMUSER.SPEC.**	IBMUSER	DSN	DAYS	N	N	31/12/1999	0	50	5	HOME		
CONT:-													
JOB NAME	DATA SET NAME	2ndVRS	2ndNAME	FSEQ	DSEQ	VOLUME	VSEQ	OWNER	CURRENT	REQUIRED	PRTY	RETDTE	RETNAME
	IBMUSER.SPEC.EXP.EXAMPLE			1	0	000010	1	IBMUSER	SHELF	SHELF	5000	25/11/2011	*
CONT:-													
NUMBER OF DATA SETS RETAINED (GROUP STORE) =										1	0		
	IBMUSER.SPEC.NOEXP.EXAMPLE			1	0	000011	1	IBMUSER	SHELF	SHELF	5000	25/11/2011	*
CONT:-													
NUMBER OF DATA SETS RETAINED (GROUP STORE) =										1	0		

5.3 测试结果进阶分析

通过对以上报告的认真分析，虽然可以看到我们新增的 VRS POLICY 对系统中数据集的影响，但在实际环境中，RMM 系统中往往存在大量的 VRS POLICY，又或者你这次增加了不只一个 VRS 定义，在这种情况下，分析以上报告将不再是一件容易的事情。

有什么更加简便易用的方法呢？在‘VERIF’的 HSKP 作业中我们看到，还有一个名为 ACTIVITY 的输出，接下来就从它入手进行分析。ACTIVITY 是针对本次 HSKP 操作的输出报告，它的数据格式被定义在系统数据集 SYS1.MACLIB(EDGACTRC)中。我们可以参照里面的数据定义编写 DFSORT 作业，将 ACTIVITY 数据集作为输入，整理出我们需要的数据信息。以下 JCL 可供参考：

```
//STEP1 EXEC PGM=ICETOOL,REGION=5M
//TOOLMSG DD DUMMY
//DFSMSG DD DUMMY
//TOOLIN DD *
COPY FROM(ACTIVITY) USING(VRST)
SORT FROM(VRST) TO(SRTDVRST) USING(SRTV)
SORT FROM(RETD) TO(SRTDRETD) USING(SRTD)
SORT FROM(MTCH) TO(SRTDMTCH) USING(SRTM)
SORT FROM(SUBC) TO(SRTDSUBC) USING(SRTC)
*
DISPLAY FROM(SRTDVRST) LIST(VRS) -
  TITLE('Data Sets Changed VRS Status') DATE TIME PAGE -
  BLANK -
  BTITLE('Status Change and Drop Reason:') BREAK(444,22,CH) -
  HEADER('DSNAME') ON(9,44,CH) -
  HEADER('JOBNAME') ON(53,8,CH) -
  HEADER('VOLSER') ON(61,6,CH) -
  HEADER('O-ST') ON(187,1,CH) -
  HEADER('N-ST') ON(188,1,CH) -
  HEADER('RSN') ON(189,1,CH) -
  HEADER('PRIMARY VRS') ON(332,44,CH) -
  HEADER('JOB MASK') ON(376,8,CH) -
  HEADER('TYPE') ON(331,1,CH)
*
OCCUR FROM(SRTDVRST) LIST(VRSS) -
  TITLE('Data Set VRS status change summary') -
  DATE TIME PAGE -
  BLANK -
  HEADER('Status Change') ON(444,9,CH) -
  HEADER('Drop Reason') ON(453,13,CH) -
  HEADER('COUNT') ON(VALCNT)
*
DISPLAY FROM(SRTDRETD) LIST(RETDATE) -
  TITLE('Data Sets Changed Retention Date') DATE TIME PAGE -
  BLANK -
  BTITLE('NEW RETENTION DATE:') BREAK(208,10,CH) -
  HEADER('DSNAME') ON(9,44,CH) -
  HEADER('JOBNAME') ON(53,8,CH) -
  HEADER('VOLSER') ON(61,6,CH) -
  HEADER('PREVIOUS') ON(198,10,CH) -
  HEADER('NEW DATE') ON(208,10,CH) -
  HEADER('PRIMARY VRS') ON(332,44,CH) -
  HEADER('JOB MASK') ON(376,8,CH) -
  HEADER('TYPE') ON(331,1,CH) -
  HEADER('SUBCHAIN') ON(400,8,CH)
```

```

*
OCCUR FROM(SRTDRETD) LIST(RETDS) -
  TITLE(' Summary of new Data Set retention dates') -
  DATE TIME PAGE -
  BLANK -
  HEADER(' New Retention Date') ON(208,10,CH) -
  HEADER(' COUNT') ON(VALCNT)
*
//ACTIVITY DD DSN=RMM. ACTIVITY. OUTPUT, DISP=SHR
//VRST      DD DSN=&&TEMPV1, SPACE=(CYL, (10, 10), RLSE), UNIT=3390
//RETD      DD DSN=&&TEMPD1, SPACE=(CYL, (10, 10), RLSE), UNIT=3390
//MTCH      DD DSN=&&TEMPM1, SPACE=(CYL, (10, 10), RLSE), UNIT=3390
//SUBC      DD DSN=&&TEMPC1, SPACE=(CYL, (10, 10), RLSE), UNIT=3390
//SRTDVRST DD DSN=&&TEMPV2, SPACE=(CYL, (10, 10), RLSE), UNIT=3390
//SRTDRETD DD DSN=&&TEMPD2, SPACE=(CYL, (10, 10), RLSE), UNIT=3390
//SRTDMTCH DD DSN=&&TEMPM2, SPACE=(CYL, (10, 10), RLSE), UNIT=3390
//SRTDSUBC DD DSN=&&TEMPC2, SPACE=(CYL, (10, 10), RLSE), UNIT=3390
//HDR       DD DSN=&&TEMPH1, SPACE=(TRK, (1, 1)), UNIT=3390
//VRSTCNTL DD *
  OUTFIL FNames=HDR,
    INCLUDE=(5, 1, CH, EQ, C' H' ),
    OUTREC=(1, 67,

      68:35, 1, CHANGE=(7, C' Y' , C' BACKUP' , C' N' , C' ' ),
      75:36, 1, CHANGE=(7, C' Y' , C' DSTORE' , C' N' , C' ' ),
      82:37, 1, CHANGE=(7, C' Y' , C' EXPROC' , C' N' , C' ' ),
      89:38, 1, CHANGE=(7, C' Y' , C' RPTXT' , C' N' , C' ' ),
      96:39, 1, CHANGE=(6, C' Y' , C' VRSEL' , C' N' , C' ' ),
      102:40, 1, CHANGE=(7, C' Y' , C' VERIFY' , C' N' , C' ' ),
      109:41, 1, CHANGE=(5, C' Y' , C' DATE' , C' N' , C' ' ),
      114:42, 1, CHANGE=(8, C' A' , C' AMERICAN' ,
        C' E' , C' EUROPEAN' ,
        C' I' , C' ISO' ,
        C' J' , C' JULIAN' ),
      122:52, 1, CHANGE=(8, C' I' , C' INFO' ,
        C' V' , C' VERIFY' ),
      130:67, 1, CHANGE=(4, C' F' , C' FAIL' ,
        C' W' , C' WARN' ,
        C' I' , C' INFO' ),
      134:68, 1, CHANGE=(3, C' N' , C' NEW' ,
        C' O' , C' OLD' ))

  OUTFIL FNames=VRST,
    INCLUDE=(5, 1, CH, EQ, C' D' , AND, 179, 1, CH, EQ, C' Y' ),
    OUTREC=(1, 357,

      444:187, 2, CHANGE=(9, C' NY' , C' RETAINED' , C' YN' , C' DROPPED' ),
      453:189, 1, CHANGE=(13, C' W' , C' WHILECATALOG' ,
        C' ' , C' ' ,
        C' U' , C' UNTILEXPIRED' ,
        C' C' , C' CYCLES' ,

```

```

C' D' , C' DAYS' ,
C' L' , C' LASTREFDAYS' ,
C' X' , C' EXTRADAYS' ,
C' B' , C' BYDAYSCYCLE' ,
C' N' , C' NO MATCH' ,
C' G' , C' DUPL. GDG' ,
C' V' , C' VOL RELEASED' ))

OUTFIL FNames=RETD,
  INCLUDE=(5, 1, CH, EQ, C' D' , AND, 180, 1, CH, EQ, C' Y' )
OUTFIL FNames=MTCH,
  INCLUDE=(5, 1, CH, EQ, C' D' , AND, 181, 1, CH, EQ, C' Y' )
OUTFIL FNames=SUBC,
  INCLUDE=(5, 1, CH, EQ, C' D' , AND, 182, 1, CH, EQ, C' Y' )
OPTION VLSHRT
//SRTVCNTL DD *
  SORT FIELDS=(187, 3, CH, A)
//SRTDCNTL DD *
  SORT FIELDS=(208, 10, CH, A)
//SRTMCNTL DD *
  SORT FIELDS=(331, 53, CH, A)
//SRTCCNTL DD *
  SORT FIELDS=(400, 32, CH, A)
/*
//VRS      DD SYSOUT=*
//VRSS     DD SYSOUT=*
//RETD     DD SYSOUT=*
//RETDS    DD SYSOUT=*

```

通过执行以上 JCL，我们可以清晰的查看到如下测试结果：

- (1) VRS，显示了本次 VRS 变更对哪些数据集产生了影响及结果

Status Change and Drop Reason: RETAINED				
DSNAME	JOBNAME	VOLSER	O-ST	N-ST
IBMUSER.SPEC.EXP.EXAMPLE		000010	N	Y
IBMUSER.SPEC.NOEXP.EXAMPLE		000011	N	Y

- (2) VRSS，显示了本次 VRS 变更对数据集影响的数量

Status Change	Drop Reason	COUNT
RETAINED		2

- (3) RETDATE，显示了本次 VRS 变更所改变的、数据集新的到期日

NEW RETENTION DATE: 25/11/2011			
DSNAME	JOBNAME	VOLSER	PREVIOUS
IBMUSER.SPEC.EXP.EXAMPLE		000010	
IBMUSER.SPEC.NOEXP.EXAMPLE		000011	

(4) RETDS, 以 NEW RETENTION DATE 为单位的数据集数量统计报告

New Retention Date	COUNT
25/11/2011	2

5.4 小结

通过查看 ACTICITY 的输出结果，我们可以非常简单、直观的判断出新增 VRS POLICY 的正确性。在确认无误后就可以执行 ‘VRSEL’ 的 RMM HSKP 作业，从而真正实现新增 VRS POLICY 对数据集的保护。


二、如何预见未来

很多使用 VRS 的用户都会问到一个问题：是否能在当日就知道明天或者一周后，RMM 系统中还会存在哪些被 VRS POLICY 保护的数据集？答案是肯定的。这绝不是通过修改系统日期实现的，也不用分析系统中数量繁多的 VRS POLICY，更不需要《预见未来》里尼古拉斯·凯奇的超能力，它只需指定一个 RMM HSKP UTILITY 参数— DATE(+n)。比如我想了解一周后的情况，那么就指定 DATE(+7)。

我们仍然以本文中数据举例说明，在“5.2 测试结果初步分析”部分的 REPORT 中，可以看到今天的日期是神棍节，即 2011 年 11 月 11 日，而数据集 IBMUSER.DAYS.NOEXP 因为匹配 IBMUSER.DAYS.**这条 VRS POLICY，它的保存日期是 2011 年 11 月 12 日。换言之，如果指定 DATE(+1)，在 HSKP 的 REPORT 输出结果中应该不会看到数据集 IBMUSER.DAYS.NOEXP 的身影。

提交以下 VERIFY 作业：

```
//EDGHSKP1 EXEC PGM=EDGHSKP,
//          PARM='VERIFY,DATE(+1)'
//SYSPRINT DD SYSOUT=*
//MESSAGE  DD DSN=RMM.MESSAGE.OUTPUT,DISP=SHR
//ACTIVITY DD DSN=RMM.ACTIVITY.OUTPUT,DISP=SHR
//REPORT   DD DSN=RMM.REPORT.OUTPUT,DISP=SHR
```

结果真的如我们所预想的吗？留给大家去预见未来吧.....

打造自己的 CPU 使用率监控工具



■ 文/ 百硕高级工程师 陈银波

看腻了 TSO RMF MONITOR III 的 CPU 使用率检查界面？在主机 SD.DA 界面，不停按回车实时看 CPU 使用率，嫌麻烦？想记录主机 CPU 秒级的实时使用率？如果你也有以上这些想法，那么接下来，笔者将为大家介绍如何利用主机 JAVA、RMF MINOTR II 接口、开放 MYSQL 数据库、PHP 网页技术、网页 AJAX 技术和 Apache web 工具，打造属于自己的 CPU 使用率监控工具。实施后，你将能通过网页看到如下 CPU 实时监控的画面：



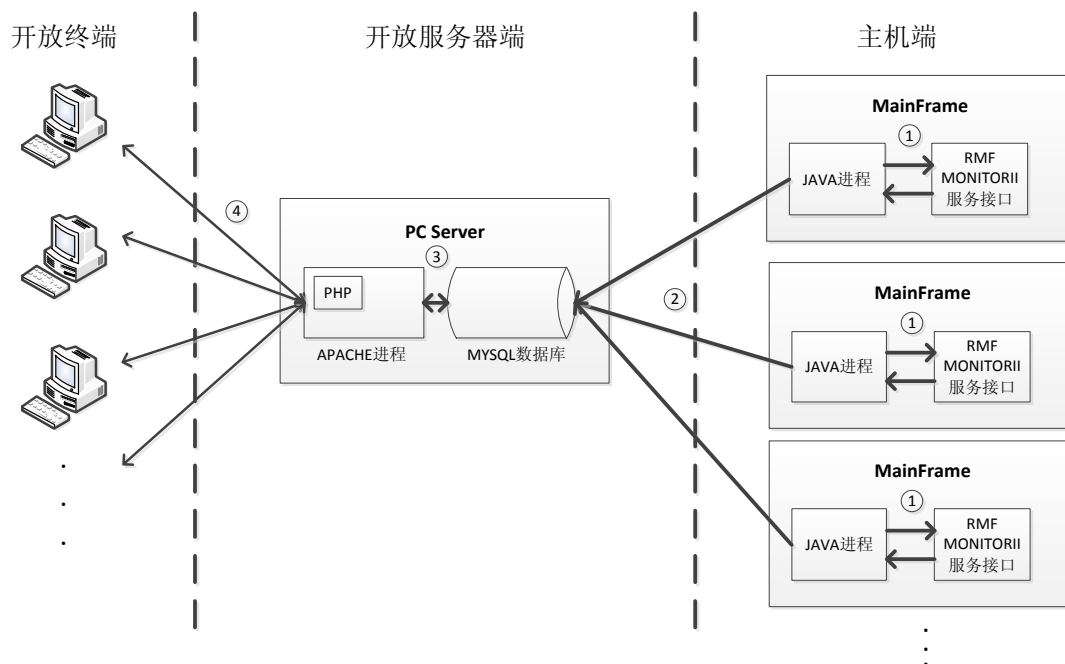
注意：上图中删除了 IP 地址和系统名等敏感信息。图表内容是利用 AJAX 技术，后台访问服务器动态更新的。

本文将给大家介绍该工具的架构、实现它所需的软件及相关技术、开发中可能碰到的难点以及主要程序的逻辑和源代码。希望在看完本文之后，对你有所启发并能制作出自己的工具。

一、架构设计

首先，让我们大致了解该 CPU 使用率监控工具的架构：JAVA 服务进程每隔 2 秒调用 RMF

MONITOR II 接口实时获取 CPU 使用率，然后通过主机 JAVA 和 MYSQL 的连接，将 CPU 数据插入到 MYSQL 数据库。同时，用户打开的在线监控页面，每隔 5 秒就会通过 Apache 服务器自动访问 MySql，将 CPU 数据获取并显示在图表中，这样就实现了对主机 CPU 使用率的在线监控。该架构如下图所示：



注解：

① JAVA 通过 JNI 方法调用 C 程序对 RMF MONITORII 接口进行访问

②通过 MySQL java JDBC Driver 将主机 JAVA 连接到 MySQL，主机 JAVA 利用这个连接完成对 MySQL 的访问

③WEB 服务器对 Mysql 的访问

④开放终端通过网页访问 Apache 服务

从图中可以看出，该架构支持多对多监控，即一个开放终端可以同时监控多个主机系统，一个系统也可以被多个终端同时监控。另外，SYSPLEX 中只要一个 LPAR 调用 RMF MONITOR II 接口就能获取所有系统的 CPU 数据。

主机服务程序使用 JAVA 语言开发，主要考虑到它的跨平台性。与 MYSQL 进行连接时就可以直接使用 JAVA 相应的驱动类，方便程序开发。

数据库选择 MYSQL，因为它不仅是免费的，而且性能也不错。另外，也可以使用 ORACLE，SQL SERVER 代替。

Web 服务器选择 APACHCE，是因为它也是免费的，而且没有那么复杂的配置过程，易学易用。另外，也可以使用 IIS 或者其他 WEB 服务器代替。

二、软件及技术需求

这里主要罗列了笔者在开发这个工具时使用到的软件和技术。

需要以下软件：

- MySQL 5.1 以上版本软件
- Apache 2.0 以上版本软件
- PHP 5.3 以上版本软件
- Java MySqlConnection 最新版本软件
- 基于 Jscript 的 FLOT 网页图表绘制库
- PHP 开源 MVC 开发框架，CodeIgniter 2.0.3
- 主机 JAVA 6.0 版本软件
- 主机 JZOS 2.10 版本软件

涉及以下计算机技术：

- 主机 RMF MONITOR II 接口访问
- HTML 网页语言，C 语言，JAVA 语言，PHP 语言，SQL 语言
- JAVA JNI 接口技术
- MySQL 数据库使用技术
- Apache 配置、PHP 配置
- AJAX 网页后台数据更新技术

三、 技术难点

在整个工具开发中，有 3 个主要技术难点：

1. 如何通过 JAVA 调用 RMF MONITOR II？

从 Java 1.1 开始，Java 就提供了 Java Native Interface (JNI)标准，它允许 Java 代码和其他语言写的代码进行交互，让 Java 具有调用本地由其他语言编译的动态库的能力。IBM 主机 JZOS 中，有些 CLASS 提供的系统功能，比如 CATALOG 访问，LOGSTREAM 访问，就是通过 JNI 技术实现。

因此，我们可以通过 C 语言编写一个动态库，在动态库中实现对 RMF MONITORII 接口的调用。然后，JAVA 再通过 JNI 技术调用这个动态库，最终实现 JAVA 调用 RMF MONITORII 接口，获得系统 CPU 实时使用率数据的功能。

2. 如何让主机 JAVA 能够直连 MYSQL 数据库？ASCII、EBDCIC 码制如何转换？

这个难点是整个工具的核心问题。主要在于连接 MySql 的 JDBC 类是需要 ASCII 编码环境的，而主机 JAVA 默认却是运行在 EBCDIC 编码环境下，在这种情况下自然无法进行连接。这种由于码制不同而引起的兼容性问题，IBM 提供了一个解决方案，即：让主机 JAVA 运行在 ASCII 编码环境下，但在读写主机 MVS 文件和数据集时进行 EBCDIC 转化。这样就很好的解决了大部分由于编码不同而引起的兼容性问题，目前他们的 Websphere 就是以这种方式运行的。按照这个解决方案，在这个工具中，对应的注意点如下：

- (1) 修改 JVM 运行作业中的环境参数，加入-Dfile.encoding=ISO8859-1，指定 JVM 运行在 ASCII 编码环境。例如：

```
IJO="-Dfile.encoding=ISO8859-1"
```

```
export IBM_JAVA_OPTIONS="$IJO "
```

- (2) 在主机 JAVA 程序中, 使用 JZOS 的 FileFactory 类读写 MVS 文件。在 ASCII 编码环境中, 当我们读写 MVS 文件时, 该类能够自动将 EBCDIC 码转换成 ASCII 码, 或者将 ASCII 码转换成 EBCDIC 码。
- (3) 通过 RMF MONITORII 接口获取的字符串数据也是 EBCDIC 码, 它可以通过 JAVA 语句【String(buff,0,len,"cp1047")】方式, 把 buff 中的数据从 EBCDIC 格式转换成 ASCII 格式。

3. 如何在网页中绘制图表, 并能自动更新?

开源的基于 JavaScript 语言的图表绘制函数库--Flot, 它能够实现在网页中绘制类似 EXCEL 中的曲线图, 而且性能较好。

网页 AJAX 技术, 可以使页面在不刷新的情况下后台访问网站获取数据, 从而更新页面显示, 让网页看起来更像一个应用程序。

解决以上三点技术问题之后, 除了安装一些必要的工具之外, 那么该工具也就能够实现了。

四、主要程序逻辑及编码

该工具总体可以分为数据获取、数据存储和数据展现三部分。数据获取部分主要为主机 JAVA 程序实现; 数据存储主要为 Mysql 数据库实现; 数据展现部分通过 PHP 网页程序实现。

● 数据获取

主机程序:

➤ C 程序 ERBMII

该程序将调用 RMF MONITOR II 接口, 然后将数据返回。程序中的头文件【ERBDSMP0】可以在 SYS1.SAMPLIB 中找到。源代码请访问如下连接获取:

<http://cinbo.blog.163.com/blog/static/14441317620119182754273/>

➤ C 程序 RmfMIICallServer.c,

该程序和 ERBMII 合并成一个动态模块被 JAVA 调用。其中的 RmfMIICallServer.h 头文件是 javah 产生的头文件。源代码请访问如下连接获取:

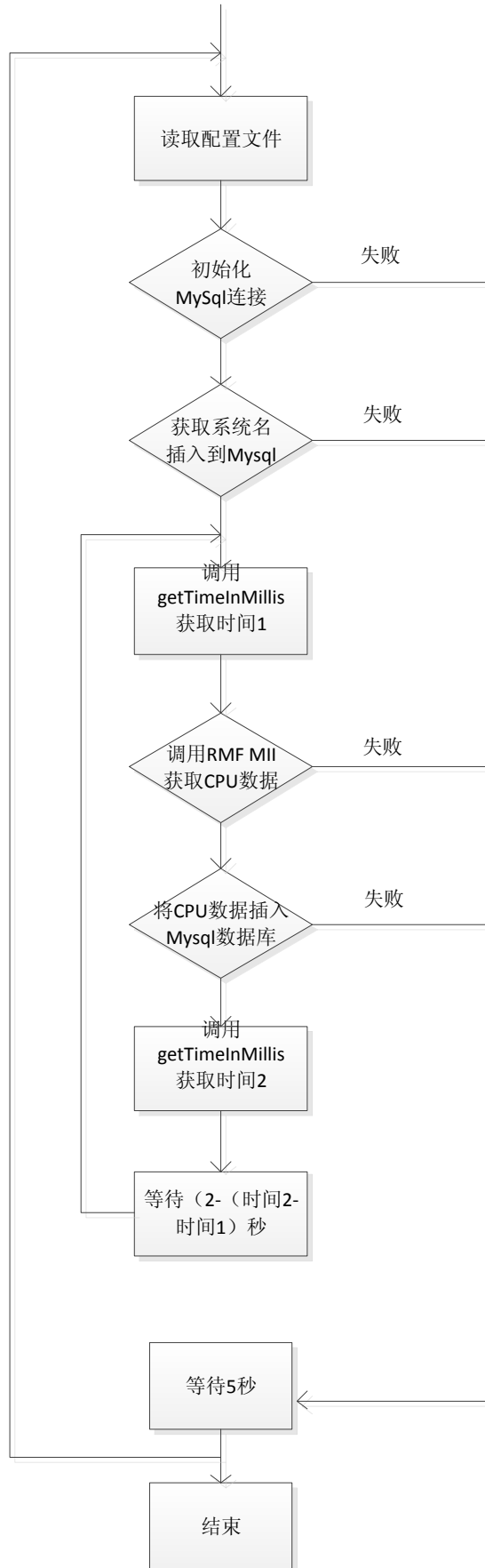
<http://cinbo.blog.163.com/blog/static/144413176201191821624353/>

➤ JAVA 主程序 RmfMIICallServer.java,

该程序为主服务程序, 它将完成读取 CPU 数据, 定时将 CPU 数据插入 Mysql 的任务。源代码请访问如下连接获取:

<http://cinbo.blog.163.com/blog/static/144413176201191822819474/>

该程序的主要执行逻辑如下:



该程序设计上将会一直运行，直到你使用/P jobname，或者 cancel 等命令。

以上三个程序共同组成了数据获取部分。

程序编译，以下命令均在 OMVS 命令行执行：

- 1、编译 java 主程序，使用命令：

```
javac RmfMIICallServer.java
```

- 2、产生 JNI 头文件，使用命令：

```
Javah -o RmfMIICallServer.h RmfMIICallServer
```

- 3、编译 ERBMII 文件，通过 JCL 作业将该程序编译成 OBJ 格式，并使用拷贝命令将 OBJ 文件拷贝到 OMVS 目录下，拷贝命令如下：

```
cp "///temp.c.obj(erbmii)" erbmii.o
```

- 4、编译 RmfMIICallServer.c 文件，使用命令如下：

```
c++ -c -I. -I/testjava/java/J6.0/include -W "c,langlvl(extended)" -W  
"c,float(ieee)" -W c,dll -W c,exportall RmfMIICallServer.c
```

其中/testjava/java/J6.0/include 目录，请根据 JAVA 实际安装目录修改。

- 5、编译生成 JNI 调用模块，该命令如下：

```
c++ -o libRmfMIIJNI.so /testjava/java/J6.0/bin/j9vm/libjvm.x  
RmfMIICallServer.o erbmii.o
```

其中/testjava/java/J6.0/bin/j9vm/libjvm.x，请根据 JAVA 实际安装目录修改。

如何运行 JAVA 主程序

请使用 JZOS 的批量方式运行主 JAVA 程序，并且加入必要的运行参数“-Dfile.encoding=ISO8859-1”

● 数据存储

当 JAVA 主程序运行时，首先自动连接到 MySQL 中的 performance_db 数据库，然后获取系统名并插入到表【sysname_list】，接着开始循环获取 CPU 使用率数据，将数据存储到表【cpu_usage】。因此需要在 Mysql 中定义 performance_db 数据库，并在该数据库下定义以下两表，完成对数据进行的存储：

表一、存放系统名称及别名

```
CREATE TABLE IF NOT EXISTS `sysname_list` (  
  `numid` int(11) NOT NULL AUTO_INCREMENT,
```

```

`SYSNAME` char(8) NOT NULL,
`PLEXNAME` char(8) NOT NULL,
`ALIAS` char(16) NOT NULL DEFAULT 'SYSALIAS',
PRIMARY KEY (`numid`),
UNIQUE KEY `SYSNAME` (`SYSNAME`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312 AUTO_INCREMENT=97 ;

```

表二、存放 CPU 数据

```

CREATE TABLE IF NOT EXISTS `cpu_usage` (
`NUMID` int(10) unsigned zerofill NOT NULL AUTO_INCREMENT,
`MVSTIME` datetime NOT NULL,
`LPARBUSY` smallint(4) NOT NULL DEFAULT '-1',
`MVSBUSY` smallint(4) NOT NULL DEFAULT '-1',
`WEBSRVTIME` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
`SYSNAME` char(4) NOT NULL,
`PLEXNAME` char(8) NOT NULL,
PRIMARY KEY (`NUMID`)
) ENGINE=MEMORY DEFAULT CHARSET=gb2312 AUTO_INCREMENT=2798607 ;

```

● 数据展现

数据展现部分是通过 PHP 网页技术、AJAX 技术和 FLOT 绘图库，将保存在数据库中的 CPU 使用率数据，实时的显示给用户。但在实际开发中，可以根据实际掌握的技术来调整，并非一定要使用 PHP 和 FLOT 绘图库。JSP、ASP 和其他绘图库也可。

关于 PHP 页面的开发，笔者使用的是 CodeIgniter 提供的 MVC 开发框架（开源的），所有 PHP 页面都有框架特性，因此在这就不再罗列 PHP 页面代码。

五、性能及资源消耗

按照设计，每个系统的 CPU 使用率数据必须每隔 2 秒被插入到 Mysql 数据库，为了尽量达到这个要求，我们需要在主机端和开放端进行一些设置：

- 将 JAVA 进程的 WLM Server Class 优先级设为 1，保证运行 JAVA 进程和调用 RMF MONITOR II 接口时，减少延时
- MySQL 数据库存放 CPU 数据表格的存储引擎使用 MEMORY 格式，保证数据从主机 JAVA INSERT 数据时，减少延迟

- CPU 数据表格每天 12 点删除前天的数据，减少表格内容，以加快 INSERT 和减少延时
- 确保 MainFrame 到 PC SERVER 的网络通畅

即使通过以上几项设置，工具在实时显示方面有时还会发生延迟，但这种情况并不影响资源监控，因此是可以接受的。

测试情况如下：

测试环境：

主机	微机
硬件：Z10, CPU 2097-707,1CP online 系统：Z/OS 1.9	硬件：XSERIES 260, CPU XEON, 8 核 系统：windows 2003 Server

测试结果：

序号	监控系统数	系统监控 页面数	插入数据 消耗时间	微机 CPU 使用率	主机进程 CPU 使用率
1	10	0	30ms	1%	0.19%
2	10	14	74ms	8%	0.19%
3	10	28	114ms	20%	0.19%
4	10	32	140ms	35%	0.19%
5	10	46	145ms	50%	0.19%


备注：

每个页面只监控一个系统，监控页面数大于系统数量，表示有多个页面同时监控了一个系统。

监控过程中，监控页面运行正常，CPU 使用率能够及时显示。

六、结束语

以上就是笔者对该工具开发的大致介绍。这是一个对利用现有计算机技术提升主机管理的探索，我们还可通过开发此类工具来学习主机 JAVA、WLM 设置、主机开放交互等技术，为将来进行技术储备。

若您对此感兴趣，欢迎与笔者联系进一步探讨，邮件请至：chen_yb@bayss.com 

清理 GLOBAL TSQ 的 几种方法



■ 文/ 百硕工程师 陆书博、王军波

一、背景

TSQ(Temporary Storage Queue)是 CICS 中用来存储临时数据的队列。TSQ 不必预先在系统中定义，在程序第一次向 TSQ 写入数据时由 CICS 自动创建。

SYSPLEX/CICS PLEX 架构下我们主要使用存放在 CF 中的 TSQ，即 Global TSQ。系统不会自动删除 TSQ，因此我们需要定期做手动清理，以释放 CF 空间。

二、清理方法

2.1 使用 MVS 系统命令

```
SetXCF Force,Structure,Strname=(DFHXQLS_poolname)
```

该命令将存放 TSQ 的 CF Structure 数据清空。

2.2 使用 CICS 交易 CEMT

```

      _ALL_____
CEMT Set TSqueue_|_____||_POOLNAME_(value_)_____DELETE_X
      |_(value_)_|          |__LASTUSEDINT__(value_)_|

```

‘Lastusedint’代表 TSQ 未被访问的时间，单位为秒。

2.3 使用 CICS SPI 命令

用户可以通过两种途径来使用 SPI 命令：

1) 用 CICS 交易 CECI 执行 SPI 命令（‘INQUIRE TSQ’和‘SET TSQ’）

当要删除的 TSQ 数量不多时、可使用此方法。

2) 在 COBOL 程序中使用 SPI 命令

核心代码如下：

```

EXEC CICS INQUIRE TSQNAME POOLNAME(T_POOL) START
EXEC CICS INQUIRE TSQNAME(T_NAME) NEXT LASTUSEDINT(T_LA) TRANSID(T_TRAN) FLENGTH(T_LEN)
EXEC CICS SET TSQNAME(T_NAME) POOLNAME(TSQ1) DELETE

```

首先使用命令‘INQUIRE START’对 TSQ 进行定位，然后循环调用‘INQUIRE NEXT’遍历所有 TSQ，从而得到 TSQ 的创建交易 ID 和长度等属性信息。

如果某个 TSQ 满足删除条件，就可调用‘SET TSQ’命令将其删除。

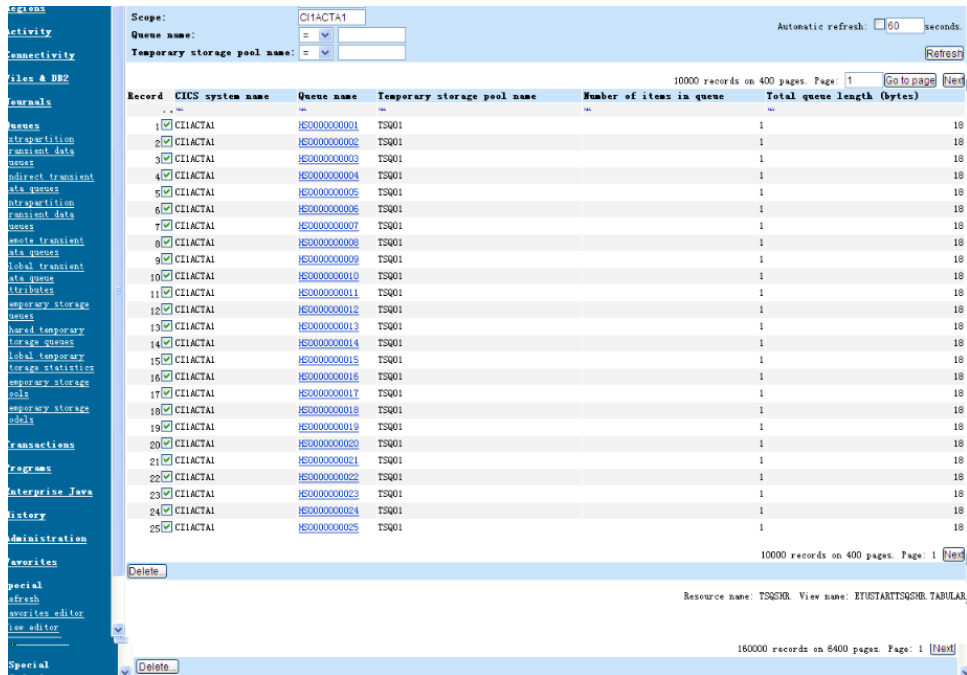
‘INQUIRE NEXT’命令在遍历到 TSQ 结尾时，会产生 END CONDITION，用户需要在 COBOL 程序中处理该 CONDITION。

2.4 使用 CPSM API

用户可以通过两种途径来使用 CPSM API:

1) 在 WUI 界面中操作

示例如下:



2) 在 REXX 程序中调用 CPSM API

核心代码如下:

```

EYUAPI ('CONNECT' ,
' VERSION (' W_VERSION' ) CONTEXT (' W_CONTEXT' )
SCOPE (' W_SCOPE' )' ,
' THREAD (W_THREAD) RESPONSE (W_RESPONSE) REASON (W_REASON)' )

EYUAPI ('PERFORM OBJECT (TSQSHR)' ,
' ACTION (DELETE)' ,
' PARM (W_PARM)' ,
' PARMLen (' W_PARML' )' ,
' CRITERIA (W_CRIT)' ,
' LENGTH (' W_CRITL' )' ,
' COUNT (W_COUNT)' ,
' RESULT (W_RESULT)' ,
' THREAD (W_THREAD)' ,
' RESPONSE (W_RESPONSE)' ,
' REASON (W_REASON)' )
    
```

REXX 中先使用 CONNECT 命令和 CMAS 建立连接，之后使用 PERFORM OBJECT 命令删除满足筛选条件的 TSQ。PERFORM OBJECT 命令的操作对象是资源表(Resource Table)，每种 CICS 资源都有对应的资源表，例如，Global TSQ 对应的资源表名称为‘TSQSHR’。用户可查阅 IBM 手册<<CICS Transaction Server for z/OS CICSplex SM Resource Tables Reference>>以获得更多关于资源表的信息。

需要注意的是，连接 CMAS 时 SCOPE 参数应设置为某个 CICS，如设置为 CICSplex，CPSM 会检索同一资源表在每个 CICS 中的实例，带来额外处理时间。

REXX 程序中的 CPSM API 命令由 CPSM 提供的 REXX 函数包（function package）负责解析并执行，因此 REXX 程序在运行时必须能够访问 CPSM 系统程序 EYU9AR00、EYU9AR01 和 IRXFLOC。这三个程序存放在 CICS 目标库（Target Library）SEYUAUTH 中。

三、优缺点对比

- 使用 MVS 系统命令直接清理 CF Structure

- ✓ 优点：简洁高效。
- ✓ 缺点：无法指定筛选条件，所有 TSQ 都会被删除。

- 使用 CICS 交易 CEMT

- ✓ 优点：简洁高效，同时支持筛选条件。
- ✓ 缺点：由于产品限制，当满足删除条件的 TSQ 数量超过 32767 时，交易执行会失败。

- 使用 CICS SPI 命令

- ✓ 优点：执行效率高、支持筛选条件、没有数量限制。
- ✓ 缺点：需要在 CICS 中部署 COBOL 程序。

- 使用 CPSM API

- ✓ 优点：支持筛选条件，REXX 程序部署较为容易。
- ✓ 缺点：无论是在 WUI 界面还是在 REXX 程序中执行，CPSM API 的效率都远低于 SPI 命令，因此只适用于要删除的 TSQ 数量较少的场合。

四、附录：测试结果

1. 使用 COBOL 程序清理 50 万个 TSQ:

测试场景	交易执行时间(秒)
查询 TSQ	19
查询并删除 TSQ	20.2
查询并打印 TSQ 信息	21.2
查询、删除并打印 TSQ 信息	24.2

2. 使用 REXX 程序清理 38 万个 TSQ:

测试场景	交易执行时间(秒)
查询 TSQ 信息	750
删除 TSQ 信息	1.3



OMEGAMON For MVS Command 使用简介



■ 文/ 百硕工程师 许扬

IBM Tivoli OMEGAMON 的作用是通过监控和管理操作系统，以及对企业应用各个组件进行可用性监控和性能分析。使用 OMEGAMON 产生的监控数据和报告，可以查看系统运行状态、跟踪处理问题、完成以下任务：

- 可视化系统实时监控
- 预设条件资源状态监控，例如设定高 CPU 使用率，应用不可用等条件
- 设定性能阈值，超过阈值后自动发出警报信息
- 跟踪导致警报的原因
- 使用“采取操作”功能，发出相应的管理指令
- 生成系统监控报告
- 客户化监控代理，根据监控需求定义监控内容

本文将阐述“可视化系统实时监控”的功能，在 3270 界面下，我们可以使用 OMEGAMON For MVS 的指令，主动监控系统和应用的健康状况，从而做出合理判断，排除问题。

下面就来介绍 OMEGAMON For MVS Command。

OMEGAMON For MVS 有以下 4 种指令类型：

指令类型	描述
INFO-line commands	INFO-line commands 执行一些控制命令，如打印屏幕 (/PRINT)或停止 OMEGAMON 的 Session(/STOP)等。INFO-line commands 需要在 Major、Minor 或 Immediate commands 这几类命令之前执行，它的不同之处在于执行后将不再显示，因此无法在屏幕上保存这类指令。INFO-line commands 通常以(/)作为开始符号，起始位置需从第二列开始。若在自动模式下，需将鼠标放在第一列的位置上。

Major commands	Major commands 用于显示基本的分类信息，如系统信息、资源利用率或内存等。在 INFO-line 下的任意行都可以输入 Major commands。
Minor commands	Minor commands 用于显示 Major commands 所选择的详细信息。在 INFO-line 下的任意行都可以输入 Minor commands，但是这些命令只在正确的 Major commands 下才会被执行。
Immediate commands	Immediate commands 提供了很多种功能：系统监控指令、查询指令以及类似 INFO-line commands 的指令。在 INFO-line 下的任意行都可以输入 Immediate commands，也可在 Major commands 及该指令自己的 Minor commands 之间执行 Immediate commands。

示例如下：

```

1 /PRINT_ _ _ _ _ #01 VTAM OM/DEX V750./C A083 March 2000 17:37
2 DISK  VMXA04  VMXA05  VMSP50  VMHP02  OMONVM  DOSTST  DP215R  +
3 dadr   1A0    1A1    1B0    1B1      2A7    2B0    4F1
4 .MIN   DADR   DALC   DIO    DIOQ    DOPN   DRES   DSTA   DTYP   DUSR  +
    
```

注解：

- 位置 1 /PRINT 是 INFO-line commands；
- 位置 2 DISK 是 Major commands，该命令显示 online disks；
- 位置 3 dadr 是 Minor commands，该命令显示 online disks 的 device numbers；
- 位置 4 .MIN 是 Immediate commands，该命令显示所有与 Major commands 有关的 Minor commands，在该示例中列出了 DISK 指令下的所有可用 Minor commands，比如除了 DADR，还有 DALC、DIO、DSTA 等指令均可使用。

案例设想：

某日，客户发现自己的 OMEGAMON 地址空间 CANSDF 在运行一段时间后，发生 S40D Abend。此时，我们可以使用 OMEGAMON MVS COMMAND 功能来查看该地址空间的内存使用情况。

首先，登录到发生问题的 CANSDF 所在 OMEGAMON MVS 地址空间，然后进入到 MVS COMMAND，接下来需要执行的命令是 PEEK。

PEEK

命令类型： OMEGAMON major command(Authorized)，执行该命令需要授权，稍后将做详细解释。

命令描述： 收集单个地址空间的信息。当用户选择使用 PEEK 来执行收集信息的指令时，有多种指令格式和显示类型适用于 PEEK 的 Minor commands。

由于 PEEK 命令需要授权，以下为与授权相关的指令：

/PWD

指令类型： OMEGAMON INFO-line command

指令描述： 指定 OMEGAMON 的密码（一般为 CANDLE3）。授权指令需要通过输入密码才能执行。

可以通过以下方法使用 /PWD command:

1. 用于内部授权，在 INFO-line 一栏输入/PWD command 即可。

示例：

Step1: /PWD_____

Step2: The system prompts you For a password.

Step3: _____ Enter Password

用户输入的 Password 不会显示在屏幕上。输入完成后请按回车键，如有“PASSWORD ACCEPTED”信息显示，再按一次以便获得所有安全命令的授权。

2. 重置安全级别，在对会话授权后将安全级别重置为 0（OMEGAMON 的安全级别分为 3 级，2 级是最高的使用权限，在此权限下，可以使用 OMEGAMON for MVS Command 中所有指令）。首先在 INFO-line 输入/PWD，不输入密码，直接按回车键。访问将会受限制，直到用户重新输入 Password。

当我们使用/PWD 指令完成对 OMEGAMON 指令的授权后，就可以使用 PEEK 命令来查看某个地址空间的信息了。PEEK 的使用格式如下：

aPEEK targ

其中“a”用于标识动作字符：

“-”用于设定从目标地址空间收集新的数据。

“<”用于设定执行前一次成功的指令。基于之前收集的数据可以继续使用 Minor commands。

当用户是初次查看 JOB 或是收集信息时，此栏位不能为空，必须输入“-”或“<”。

“targ”表示目标地址空间，允许的格式如下：

ccccccc jobname

nnnn decimal ASID number

* OMEGAMON ADDRESS SPACE

例如，我们想要收集作业名为 CANSDF（其 ASID 为 10 进制数 25），可以输入：

-PEEK CANSDF

或

```
-PEEK 25
```

按回车键后显示如下:

```
PEEK CANSDF ASID=25 >> OB8112: Data Collection Initiated <<
```

以上信息表示 CANSDF 的信息已经被收集完成, 之后可以输入 PEEK 的 Minor commands, 用于查看 CANSDF 该地址空间的详细信息:

```
Step4: AMAP
```

按回车键后, OMEGAMON 显示该地址空间内存图如下:

```
PEEK USER01 ASID=46, collected at 15:39:39
amap
+ ===== 2 Gig Line ===== <== 7FFFFFFF Highest 31-bit address
+
+ |-----| <== 7FFFFFFF Top of Extended Private
+ |//////|
+ |// System Area //| >----- 8,948K ELSQA,SWA unallocated
+ |//////| 6K Fragmented free space
+ |-----| <== 7F741000 Current bottom of ELSQA,SWA
+
+ A Available >----- 1,966M Avail. for ELSQA/SWA only
+ M
+ A |-----| <== 048FFFFFF Extended User Area Limit
+ P
+ A Available >----- 32,668K Avail. for ELSQA/SWA/USER
+
+ |-----| <== 02918FFF Current Top of Ext. User Area
+ |//////| 4K Largest free block
+ |// User Area //| >----- 4K Extended User unallocated
+ |//////| 7K Fragmented free space
+ |-----| <== 02900000 Bottom of Extended Private
+
+ ===== 16 Meg Line ===== <== 00FFFFFF Highest 24-bit address
+
+ |-----| <== 007FFFFFF Top of Private
+ |//////|
+ A // System Area // >----- 24K LSQA/SWA unallocated
+ M |//////| 42K Fragmented free space
+ A |-----| <== 007D0000 Current Bottom of LSQA/SWA
+ P
+ B Available >----- 2,796K Avail. for LSQA/SWA only
+
+ |-----| <== 00514FFF User Area Limit
+
+ Available >----- 104K Avail. for LSQA/SWA/USER
+
+ |-----| <== 004FAFFF Current top of User Area
+ |//////| 128K Largest free block
+ |// User Area //| >----- 204K User unallocated
+ |//////| 40K Fragmented free space
+ |-----| <== 00005000 Bottom of Private
+
+ === Absolute Bottom === <== 00000000 Prefixed Storage Area
```

OMEGAMON 会展示该地址空间在私有区域中虚存的使用情况。

AMAP 还能控制显示的区域，通过使用 amapA 和 amapB 命令，可分别显示 16M 线上或线下区域。其中“A”表示线上，“B”表示线下。如下图所示：

```
PEEK USER01 ASID=46, collected at 15:39:39
amap <map all virtual storage>
amapA <map virtual storage above the 16M line> (XA and ESA)
amapB <map virtual storage below the 16M line>
```

经检查，我们发现 CANSDF 在 16M 线下的内存严重不足了，需要对相应的参数进行调整。我们可以进入到数据集 OMEG.RKANPARU 中，对存放在 MEM-KDFSYSIN 中的 RESERVE 参数进行调整，在该参数中会指定预留给 CANSDF 的内存。其中参数 (2048, P) 中的“P”表示线下部分，(2048, X) 中的“X”表示线上部分。之前我们通过 AMAP 指令发现，是 Private 区域不够，因此我们将 (2048, P) 扩展到 (4096, P)。这样修改后我们再重启 CANSDF，没有再发生 S40D Abend，这表示对于参数的修改是成功的。（注意：在 OMEGAMON 该 MEM 中，RESERVE 设定的参数不能超过 MINIMUM 这一基本原则，所谓预留就是说在 MINIMUM 分配的内存范围内保留一块，是不应该超过 MINIMUM 的内存范围）。

下面是在 MAP 中对于每个栏位的具体描述。

16M 线下区域的描述：

Area descriptions for storage above the 16M line:

Highest 31-bit address	The highest possible address in 31-bit architecture.
Top of extended private	Highest address within the extended private area.
ELSQA/SWA unallocated	The amount of storage not currently allocated within the extended system area.
Fragmented free space	The amount of free storage within allocated pages of the extended system area.
Current bottom of ELSQA/SWA	Lowest address allocated within the extended private area for the extended system area.
Avail. for ELSQA/SWA only	The amount of unallocated storage between the current bottom of the extended system area and the limit of the extended user area.
Extended User Area Limit	Highest address possible for the extended user area.
Avail. for ELSQA/SWA/USER	The amount of unallocated storage between the extended user area limit and the current top of extended user area. Note that the extended system area can allocate storage within this area.
Current Top of Ext. User Area	The highest address currently allocated within the extended private area for the extended user area.
Largest free block	The largest contiguous piece of unallocated storage within the extended user area.
Extended User unallocated	The amount of storage not allocated within the extended user area.
Fragmented free space	The amount of free storage within allocated pages of the extended user area.
Bottom of Extended Private	The lowest address currently allocated within the extended private area for the extended user area.

16M 线上区域的描述：

Area descriptions for storage below the 16M line:


Highest 24-bit address	The highest possible address in 24-bit architecture.
Top of Private	Highest address below the common area (start of CSA).
LSQA/SWA unallocated	Total of contiguous 4K areas. The numbers include LSQA, SWA and subpools 229/230.
Fragmented free space	Total of areas within LSQA which are each less than the 4K available for allocation as defined by FQEs.
Current bottom of LSQA/SWA	Lowest address allocated to LSQA/SWA subpools.
Avail. for LSQA/SWA only	Total space available for LSQA/SWA allocation. This includes the LSQA/SWA unallocated value and the amount of space in the region available area.
User Area Limit	Highest address available for user allocation (region size plus IEALIMIT).
Avail. for LSQA/SWA/USER	Amount of space available for problem program allocations, not including unallocated areas within the region used.
Current top of User Area	Highest address currently allocated for problem program use.
Largest free block	Largest contiguous area available within the region used.
User unallocated	Total of the contiguous 4K areas within the region used which are available for problem program use.
Bottom of Private	Lowest address within the private area (above the resident nucleus rounded up to the next 64K boundary).
Prefixed Storage Area	Fixed storage location starting with absolute zero.

下表为 PEEK 相关的 Minor commands 及其含义：

Name	Type	Description
AMAP	Minor commands	显示 private 区域中内存使用情况。该图可显示最大 region 可用数，当前 region 使用情况以及 region 中的其他区域。
DATA	Minor commands	显示对于一个给定的地址空间的数据空间和数据空间使用情况（显示 data space 需要安装 MVS/SP 3.1，Hiperspace 需安装 DFP3）。
DDNS	Minor commands	显示 allocated ddnames。
JOBS	Minor commands	显示来自于 private 区域的可用值。
MODS	Minor commands	显示目前被 loaded into the user's job pack area 的 modules。
STEP	Minor commands	显示 private area 的内存使用情况。

SUBP	Minor commands	显示目前 virtual storage allocations for each storage Subpool。
TCBS	Minor commands	显示目前 the current TCB structure for the target user。
WSIZ	Minor commands	Alters the work area size for PEEK data。

综上所述，OMEGAMON 命令可以帮助我们在日常工作中快速对问题进行定位、分析和预防，极大地方便了对 OMEGAMON 软件的维护和操作。大家还可通过《OMEGAMON® for MVS Commands and Keywords》以获取更多相关信息。

笔者也希望和大家共同探讨实际工作中的使用心得，邮件请至 xu_yang@bayss.com。 

数据传输中 常见问题的解答



■ 文 / 百硕工程师 王晨

数据传输是日常工作中不可缺少的一个环节，笔者主要以数据传输工具---Connect:Direct (C:D) 为例，解答数据传输过程中的一些常见问题，希望能对读者有所帮助。

问题 1：生僻字转码异常

答：在开放平台和主机平台互传文件时，经常会发生源文件中因为含有生僻字，最后在目标文件中该生僻字被转换成乱码或不可见字符。

解决方法很简单，只需将该生僻字的 EBCDIC 码及 ASCII 添加至 C:D 的码表重新编译后，再传输，该问题便可轻松解决。目前 C:D 的码表包含 24000 多个字符，基本上囊括了所有中文汉字及字符。

问题 2：如何更改不同平台间转码时的字符对应关系？

答：在问题 1 中我们谈到，遇到生僻字可以修改 C:D 码表来解决，那么该如何修改这个码表？不同平台间字符的对应关系可以根据实际问题进行更改。譬如：开放平台的应用程序以“`”为分隔符将数据录入到数据库，而主机平台的源文件是以空格为分隔符，当文件从主机传到开放平台后，文件无法录入到数据库中。

我们可以将 C:D 码表中空格对应的值修改为“`”的 ASCII 码值。修改完成后再传输，该文件中的空格将被自动替换为“`”，而不需要修改开放平台相关的应用程序完成数据录入。我们不难看出，通过对 C:D 码表的修改，可以完成任意字符的对应转换，这个解决方法具有通用性。

问题 3：如何减少大文件传输前的压缩时间？

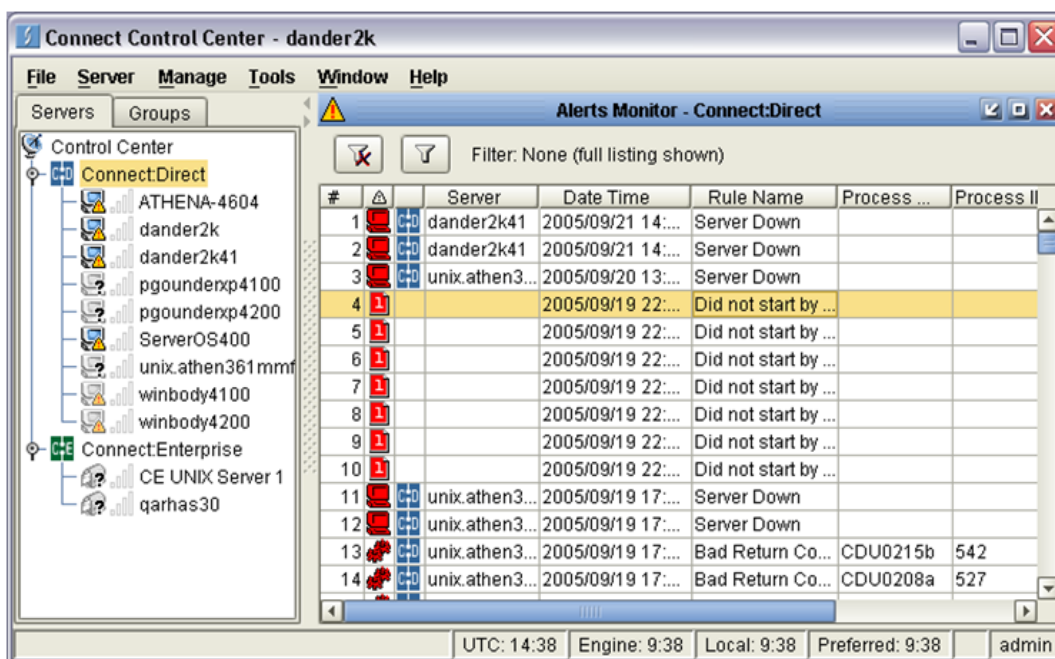
答：当传输大文件时，往往先将文件进行压缩处理后再进行，当文件完整的传输到目标环境后，还需对文件进行解压缩处理。文件越大，压缩与解压缩所消耗时间就越长。当使用 C:D 传输时，由于该产品具有压缩传输功能，它可以做到压缩、传输、解压缩同时进行，从而省去了文件传输前后的压缩与解压缩时间。

问题 4：如何隐藏传输作业或脚本中的用户名及密码？


答：当使用 FTP 传输文件时，需要将远程节点的 IP、用户名、密码以明文的形式写在作业或脚本中，这样就会带来安全隐患。当使用 C:D 工具时，远程节点 IP、用户名、密码在 C:D 产品的 NETMAP 及 USERFILE 中进行定义，无需将 IP 地址、用户名及密码写在作业或脚本中，加上 C:D 产品的 NETMAP 及 USERFILE 可以通过权限控制对其访问，从而规避了用户名、密码等明文的泄漏，提高了系统的安全性。

问题 5：如何第一时间得知文件传输发生异常？

答：在使用 FTP 传输文件时，传输结果通常要看作业或脚本的返回码进行确认。当文件传输异常时，因为某种原因没有及时发现，就可能会造成生产问题隐患。C:D 可以结合 Sterling Control Center(SCC) 监控平台使用，对所监控 C:D 节点的文件传输进程进行全程监控，并以图形化界面呈现（如下图），如发生 C:D 进程异常、文件传输异常时，SCC 会在监控界面上显示各种异常警告信息，并将这些信息通过邮件或短信的方式发送给相关负责人。



问题 6：如何实现不同传输工具或协议间的数据交互？

答：例举有数据交互需求的甲、乙、丙三家企业。甲使用 C:D、乙使用 FTPS、丙使用 MQ。由于安全管理限制，他们均不允许使用对方企业的数据传输工具。如何才能满足三家企业间的数据交互需求？IBM 数据传输整合工具---Sterling Integrator (SI) 可以做到。SI 是一个支持多协议如 HTTP、HTTPS、SMTP、FTP、FTPS、C:D、MQ、EDIINT 等的数据传输整合工具。三家企业只需将数据传输至装有 SI 的服务器，由 SI 根据预先定义的 routine 调用对应的 BP (Business Process)，再将数据发送至对应的合作伙伴，从而实现不同协议间的数据交互。 

百硕客户通讯 BAYSHORE ADVISOR

中国主机用户专享的资讯季刊

2011 年 12 月 1 日出版（总第 26 期）

本期责任编辑：马彤雷

主办：百硕同兴科技（北京）有限公司

出版：百硕客户通讯编委会

吕 宁

李 琰

王晓兵

徐卫华

邹杰

高春霞

刘京平

高大川

陈银波

马彤雷

罗兴魁

贺 明

康会影

彭晰遥

Martha Hall

地址：北京市朝阳区望京科技园利泽中二路 1 号中辰大厦 209 室

电话：010 64391733 传真：010 64391582

电子邮箱：Bayshore_advisor@bayss.com

如果您对百硕客户通讯有任何意见和建议，欢迎您随时与我们交流！



百硕同兴

百硕客户通讯总第 26 期 (2011 年 12 月 1 日)
百硕同兴科技 (北京) 有限公司
Bayshore Consulting and Service Co., LTD.