Virtualization Standards for Business Continuity: Part 9

This is the ninth of a series of articles defining the policies, guidelines, standards, and procedures that provide the foundation of a virtualized environment, thus enabling business continuity, disaster recovery, and high availability, with an emphasis toward Return On Investment (ROI).

This article will focus on a standardized procedure for building and configuring the client LPAR. The client LPAR will utilize I/O resources provided by dual VIO servers, as described in previous articles in this series. This article includes a shell script called "mklpar" that is used to create and configure client LPAR's.

The "mklpar" shell script is written in Korn Shell 93 code and uses default values for many of the client LPAR configuration parameters, however these can be modified by the user on an as needed basis. These parameters include settings for CPU, memory, physical and virtual resource sharing. As described in the previous article of this series, the users of this script should know and understand some basic Power5 and Power6 architecture. The CPU settings are divided into two categories, processing units and virtual processors.

Processing units represent the amount of physical CPU processing power that can be utilized by the client LPAR. The minimum amount of CPU processing units that can be assigned to an LPAR is 0.10 or $1/10^{th}$ of a processor, and can be incremented above that in increments of 0.01 processing units or $1/100^{th}$ of a processor. A single virtual processor is represented on an LPAR as two active logical processors on a system with symmetric multi-threading (SMT) enabled, or a single active logical processor if SMT is not enabled. Each logical processor translates into a processing thread. The reason this is important to know is the minimum number of processing units that can be assigned to a virtual processor is 0.10 or $1/10^{th}$ of a processor. This means that for every virtual processor required by an LPAR, 0.10 processing units must also be allocated. So an LPAR requiring four (4) virtual processors (8 logical processors or SMT threads) requires a minimum of 0.40 processing units.

The desired values for processing units and virtual processors on a client LPAR can be determined by the system administrator based on the expected processing load of that LPAR. Table 9.1 provides a list of suggested initial CPU parameters for a client LPAR, if the processing load is unknown or cannot be determined at the time the client LPAR is built.

Description	***mklpar Variable***	***Default Value***
Minimum Virtual Processors	MIN_PROCS	*2*
Desired Virtual Processors	DESIRED_PROCS	2
Maximum Virtual Processors	MAX_PROCS	4
Minimum Processing Units	MIN_PROC_UNITS	0.20
Desired Processing Units	DESIRED_PROC_UNITS	0.20
Maximum Processing Units	MAX_PROC_UNITS	1.00
Processing Units Sharing Mode	SHARING_MODE	ON
Processing Units Uncapped Weighting	UNCAP_WEIGHT	128

Table 9.1: CPU Parameters for client LPAR

Copyright 2007 by Dana French

The memory requires will be dependent upon the number and types of applications the client LPAR will support. These values will vary widely and should be determined before building the client LPAR. Table 9.2 provides default memory values if these are unknown or cannot be determined prior to building the client LPAR.

Description	***mklpar Variable***	***Default Value***
Minimum Memory	MIN_MEM	1024 Mb
Desired Memory	DESIRED_MEM	1024 Mb
Maximum Memory	MAX_MEM	2048 Mb

Table 9.2: Memory Parameters for Client LPAR

Virtual slot numbering standards used by the "mklpar" script correspond with the same standards used for building the VIO servers. However the client LPAR will have slots provided by both VIO servers for redundant access to storage and networking. As described in previous articles, the even numbered virtual slots will be provided by the even numbered VIO server, the odd numbered virtual slots are provided by the odd numbered VIO server. Storage redundancy is provided by the IBM Multi-Path I/O (MPIO) driver. Network redundancy can be configured by two different mechanisms from the VIO server, those mechanisms are referred to as:

- Shared Ethernet Adapter (SEA) fail over
- Multiple ethernet adapters configured as an aggregated link

The network redundancy mechanism configured by the scripts described by this set of articles is the aggregated link method, and is normally referred to as "etherchannel". This method was chosen because it provides greater flexibility, however it is more difficult to configure and requires network switches to be configured to support etherchannels for multiple physical adapters.

The "mklpar" shell script assumes the user on the system where the script is being executed, has password-less ssh access to the HMC via the "hscroot" or other administration level user ID. Managed system names can be displayed using the "-L" option of the "mklpar" script.

Physical and virtual I/O are expected to be added by the system administrator after creating the client LPAR using the "mklpar" script. This task can be automated through scripting, however it requires adherence to policies, guidelines, standards, and procedures to ensure physical and virtual I/O slots and adapters are not reused between multiple client LPARs. Keeping track of the used adapters, disks, and slots is the most difficult task associated with automating virtualized LPAR's, but it can and is done by a product called "vLPAR", search *google* for more information about this.

```
#!/usr/bin/ksh93
#################################################################
function usagemsg_mklpar {
  print "
Program: mklpar
```

Copyright 2007 by Dana French

Virtualization Standards for Business Continuity: Part 9

```
Create a template LPAR on an HMC managed system.

Usage: ${1##*/} [-?vV] -h HMCname -l LPARname [-u HMCuser]
            [-s SystemName] [-w #] [-L]
  Where:
    -v = Verbose mode - displays mklpar function info
    -V = Very Verbose Mode - debug output displayed
    -h = HMC machine name
    -u = HMC user name
    -l = LPAR name to create
    -s = System name on which to create the LPAR
    -L = Display a list of systems managed by the specified HMC
    -w = Workgroup management number (default:1)

Author: Dana French (dfrench@mtxia.com) Copyright 2005
\"AutoContent\" enabled
"
}
#####################################################################
####
#### Description:
####
#### This script will create a template LPAR on an HMC
#### managed system.  The resulting LPAR will require manual
#### configuration of physical and/or virtual I/O adapters.
####
#### Environment variables may be specified to change the default
#### values of the LPAR template.
####
#### Assumptions:
####
#### This script assumes "ssh" connectivity to the HMC, but
#### not necessarily "password-less" login.
####
#### Dependencies:
####
#### Connectivity with the HMC using "ssh" is the only
#### communication mechanism supported at this time.
####
#### Products:
####
#### The product of this script is a configured LPAR and profile
#### on an HMC managed system.  No I/O is configured thru
#### this script, that is left for manual configuration.
####
#### Configured Usage:
####
#### Environment variables that can be specified to change the
#### default values of the LPAR template:
####
####    LPAR_ENV
####    WORK_GROUP_ID
####    MIN_MEM
####    DESIRED_MEM
####    MAX_MEM
```

Copyright 2007 by Dana French

Virtualization Standards for Business Continuity: Part 9

```
####    PROC_MODE
####    MIN_PROC_UNITS
####    DESIRED_PROC_UNITS
####    MAX_PROC_UNITS
####    MIN_PROCS
####    DESIRED_PROCS
####    MIN_PROCS
####    DESIRED_PROCS
####    MAX_PROCS
####    SHARING_MODE
####    UNCAP_WEIGHT
####    BOOT_MODE
####    CONN_MONITORING
####
#### Details:
####
#####################################################################
function trap_mklpar {
  print -u 2 "# INFO: key trapped and ignored during critical phase"
  return 0
}
#####################################################################
function mklpar {
  typeset VERSION="1.0"
  typeset TRUE="1"
  typeset FALSE="0"
  typeset EXITCODE="0"
  typeset VERBOSE="${FALSE}"
  typeset VERYVERB="${FALSE}"
  typeset HMCUSER=""
  typeset HMCNAME=""
  typeset LPARNAME=""
  typeset SYSNAME=""
  typeset LPAROUT=""
  typeset HMCLIST="${FALSE}"

####
#### For each of the LPAR parameters, check to see if an environment
#### variable exists with the same name.  If so, use the value of
#### the environment variable, otherwise use the specified default
#### value.
####

  typeset LPAR_ENV="${LPAR_ENV:-aixlinux}"
  typeset WORK_GROUP_ID="${WORK_GROUP_ID:-}"
  typeset MIN_MEM="${MIN_MEM:-512}"
  typeset DESIRED_MEM="${DESIRED_MEM:-1024}"
  typeset MAX_MEM="${MAX_MEM:-2048}"
  typeset PROC_MODE="${PROC_MODE:-shared}"
  typeset MIN_PROC_UNITS="${MIN_PROC_UNITS:-0.2}"
  typeset DESIRED_PROC_UNITS="${DESIRED_PROC_UNITS:-0.2}"
  typeset MAX_PROC_UNITS="${MAX_PROC_UNITS:-1.0}"
  typeset MIN_PROCS="${MIN_PROCS:-$(( int(MIN_PROC_UNITS * 10) ))}"
  typeset DESIRED_PROCS="${DESIRED_PROCS:-$(( int(DESIRED_PROC_UNITS * 10) ))}"
  typeset MAX_PROCS="${MAX_PROCS:-$(( int(MAX_PROC_UNITS * 10) ))}"
  typeset SHARING_MODE="${SHARING_MODE:-uncap}"
```

Copyright 2007 by Dana French

Virtualization Standards for Business Continuity: Part 9

```
  typeset UNCAP_WEIGHT="${UNCAP_WEIGHT:-128}"
  typeset BOOT_MODE="${BOOT_MODE:-norm}"
  typeset CONN_MONITORING="${CONN_MONITORING:-1}"

####
#### Process the command line options and arguments, saving
#### the values as appropriate.
####

  while getopts ":vVu:h:l:s:w#L" OPTION
  do
     case "${OPTION}" in
        'u') HMCUSER="${OPTARG}";;
        'h') HMCNAME="${OPTARG}";;
        'l') LPARNAME="${OPTARG}";;
        's') SYSNAME="${OPTARG}";;
        'L') HMCLIST="${TRUE}";;
        'w') WORK_GROUP_ID="${OPTARG}";;
        'v') VERBOSE="${TRUE}";;
        'V') VERYVERB="${TRUE}";;
        '?') usagemsg_mklpar "${0}" && return 1 ;;
        ':') usagemsg_mklpar "${0}" && return 1 ;;
     esac
  done

  shift $(( ${OPTIND} - 1 ))

####
#### If the maximum number of processors or processing units
#### is greater than the maximum allowable for a frame (64)
#### then reset the maximum value to "64".
####

  (( MAX_PROC_UNITS > 64 )) && MAX_PROC_UNITS="64"
  (( MAX_PROCS > 64 )) && MAX_PROCS="64"

####
#### Verify the HMC machine name and LPAR name was specified,
#### if not display an error message and return from this script.
#### The HMC machine name and LPAR name are the only required
#### command line parameters, all others are optional.
####

  trap "usagemsg_mklpar ${0}" EXIT
  if [[ "_${HMCNAME}" = "_" ]]
  then
    print -u 2 "ERROR: HMC machine name not specified"
    return 2
  fi
  if (( HMCLIST == FALSE )) && [[ "_${LPARNAME}" = "_" ]]
  then
    print -u 2 "ERROR: LPAR machine name not specified"
    return 3
  fi

####
```

Copyright 2007 by Dana French

```
#### Check to see the MIN and DESIRED values are less than or
#### equal to the MAX values.  If not display an error message
#### and return from this script.
####

  (( MIN_PROC_UNITS > DESIRED_PROC_UNITS )) &&
     print -u 2 "ERROR: Processing Units: Min > Desired" &&
     return 4
  (( MIN_PROC_UNITS > MAX_PROC_UNITS )) &&
     print -u 2 "ERROR: Processing Units: Min > Max" &&
     return 5
  (( DESIRED_PROC_UNITS > MAX_PROC_UNITS )) &&
     print -u 2 "ERROR: Processing Units: Desired > Max" &&
     return 6

  (( MIN_PROCS > DESIRED_PROCS )) &&
     print -u 2 "ERROR: Processors: Min > Desired" &&
     return 7
  (( MIN_PROCS > MAX_PROCS )) &&
     print -u 2 "ERROR: Processors: Min > Max" &&
     return 8
  (( DESIRED_PROCS > MAX_PROCS )) &&
     print -u 2 "ERROR: Processors: Desired > Max" &&
     return 9

  trap "-" EXIT

####
#### Display some program info and the command line arguments specified
#### if "VERBOSE" mode was specified.
####

  (( VERYVERB == TRUE )) && set -x
  (( VERBOSE  == TRUE )) && print -u 2 "# Version.....: ${VERSION}"
  (( VERBOSE  == TRUE )) && print -u 2 "# HMC Name.....: ${HMCNAME}"
  (( VERBOSE  == TRUE )) && print -u 2 "# HMC User.....: ${HMCUSER:-<CURRENT USER>}"
  (( VERBOSE  == TRUE )) && print -u 2 "# System Name....: ${SYSNAME:-<NOT SPECIFIED>}"
  (( VERBOSE  == TRUE )) && print -u 2 "# LPAR Name.....: ${LPARNAME}"
  (( VERBOSE  == TRUE )) && print -u 2 "# Work Group ID..: ${WORK_GROUP_ID}"

#####################################################################

####
#### Obtain a list of managed system names from the HMC and
#### store the list in an array for later use.
####

if (( HMCLIST == TRUE )) || [[ "_${SYSNAME}" = "_" ]]
then
  SYSLIST=( $( ssh ${HMCUSER:+${HMCUSER}@}${HMCNAME} "lssyscfg -r sys -F name" ) )
fi

####
#### If the "-L" option was specified on the command line, display
#### the list of managed system names and  return from this function.
####
```

Copyright 2007 by Dana French

Virtualization Standards for Business Continuity: Part 9

```
if (( HMCLIST == TRUE ))
then
  for SYSTMP in "${SYSLIST[@]}"
  do
    print "${SYSTMP}"
  done
  return 0
fi

####
#### If the managed system name was not specified on the command line,
#### display a selection menu of managed system names, using the
#### array of names obtained earlier.  Prompt the user to select
#### a name.
####

if [[ "_${SYSNAME}" = "_" ]]
then
  PS3=$'\n"'Select a system by number: "
  print "\nSelect a system managed by the HMC \"${HMCNAME}\"”
  print “on which to create the LPAR \"${LPARNAME}\".\n"
  select SYSNAME in "${SYSLIST[@]}"
  do
    if [[ "_${SYSNAME}" != "_" ]]
    then
      print "\n# System Name Selected: ${SYSNAME}"
      break
    else
      print -u 2 "\n# ERROR: Invalid selection\n"
    fi
  done
fi

  (( VERBOSE  == TRUE )) && print "# Creating LPAR \"${LPARNAME}\""

  trap "trap_mklpar ${0}" HUP
  trap "trap_mklpar ${0}" INT
  trap "trap_mklpar ${0}" QUIT
  trap "trap_mklpar ${0}" TERM

####
#### Create the template LPAR on the managed system
####

ssh ${HMCUSER:+${HMCUSER}@}${HMCNAME} "mksyscfg -r lpar \
-m ${SYSNAME} \
-i name=${LPARNAME},\
profile_name=${LPARNAME}\
${LPAR_ENV:+,lpar_env=${LPAR_ENV}}\
${WORK_GROUP_ID:+,work_group_id=${WORK_GROUP_ID}}\
${MIN_MEM:+,min_mem=${MIN_MEM}}\
${DESIRED_MEM:+,desired_mem=${DESIRED_MEM}}\
${MAX_MEM:+,max_mem=${MAX_MEM}}\
${PROC_MODE:+,proc_mode=${PROC_MODE}}\
${MIN_PROC_UNITS:+,min_proc_units=${MIN_PROC_UNITS}}\
```

Copyright 2007 by Dana French

Virtualization Standards for Business Continuity: Part 9

```
${DESIRED_PROC_UNITS:+,desired_proc_units=${DESIRED_PROC_UNITS}}\
${MAX_PROC_UNITS:+,max_proc_units=${MAX_PROC_UNITS}}\
${MIN_PROCS:+,min_procs=${MIN_PROCS}}\
${DESIRED_PROCS:+,desired_procs=${DESIRED_PROCS}}\
${MAX_PROCS:+,max_procs=${MAX_PROCS}}\
${SHARING_MODE:+,sharing_mode=${SHARING_MODE}}\
${UNCAP_WEIGHT:+,uncap_weight=${UNCAP_WEIGHT}}\
${SHARING_MODE:+,sharing_mode=${SHARING_MODE}}\
${BOOT_MODE:+,boot_mode=${BOOT_MODE}}\
${CONN_MONITORING:+,conn_monitoring=${CONN_MONITORING}}"

####
#### Create virtual ethernet adapters
####

  (( VERBOSE  == TRUE )) && print -u 2 "#     Creating virtual ethernet adapters"

ETHAPPL0=500
ETHAPPL1=501
ETHMGMT0=850
ETHMGMT1=851

ssh ${HMCUSER:+${HMCUSER}@}${HMCNAME} "chsyscfg -r prof \
-m ${SYSNAME} \
-i lpar_name=${LPARNAME},name=${LPARNAME},max_virtual_slots=999,\
\\\"virtual_eth_adapters=${ETHAPPL0}/1/${ETHAPPL0}//0/0,\
${ETHAPPL1}/1/${ETHAPPL1}//0/0,\
${ETHMGMT0}/1/${ETHMGMT0}//0/0,\
${ETHMGMT1}/1/${ETHMGMT1}//0/0\\\""

####
#### Retrieve the LPAR information from the managed system for the
#### newly created LPAR and display this information if the "VERBOSE"
#### option was specified.
####

  if (( VERBOSE  == TRUE ))
  then
     LPAROUT=$( ssh ${HMCUSER:+${HMCUSER}@}${HMCNAME} "lssyscfg -r lpar \
        -m ${SYSNAME} \
        --filter lpar_names=${LPARNAME} -F name,lpar_id,lpar_env,state" )
     print "#     name, id, type, state"
     print "#     ${LPAROUT}"
  fi

  trap "-" HUP
  trap "-" INT
  trap "-" QUIT
  trap "-" TERM

  return 0
}
#################################################################

mklpar "${@}"
```

Copyright 2007 by Dana French

Virtualization Standards for Business Continuity: Part 9

The Business Continuity policies, guidelines, standards, and procedures to be learned from this article are as follows:

Policies: *Those tasks that must be implemented*

- Single points of failure will be eliminated in the client LPAR by utilizing resources from dual VIO servers.
- Client LPAR's with standardized configurations will be implemented for all business function requirements.
- Shared CPU and memory resources shall be configured for all new client LPAR's.

Guidelines: *Those tasks that should be implemented*

- Client LPAR's should be allocated an adequate amount of CPU and memory to provide the performance necessary to support the expected business function application load.
- CPU and Memory values may be augmented after installation based on software requirements.
- Physical I/O Adapters in production environments may be dedicated to a client LPAR based on bandwidth requirements.

Standards: *Technical specifications derived from the policies and guidelines*

- This article provides default CPU and memory settings for client LPAR's. These values should be adjusted for your organizations needs and requirements, then documented as a client LPAR standard.
- Client LPAR's utilize standardized virtual I/O slots from dual VIO servers.
- A standardized client LPAR configuration is provided in this article via a shell script named "mklpar".

Procedures: *Step-by-step implementation instructions of the standards*

- This article provides a standardized procedure for creating VIO client LPAR's. This procedure is in the form of a shell script named "mklpar".

The next article in this series will discuss standardized procedures for installing the AIX operating system on client LPAR's, configuring the network etherchannel adapters, and prioritizing the storage communication on the client LPAR between the VIO servers.

Dana French
President, Mt Xia, Inc.
http://www.mtxia.com
615.556.0456

Copyright 2007 by Dana French