(ITEA2 09011)

Optimize HPC Applications on Heterogeneous Architectures

Deliverable: D5-2.1.4 (L 3.1.4)

Demonstration of energy measurement using an accounting tool collecting the energy consumption data via the API

Version: V1.1

Date: January 2015

Authors: Bull / Anne Laverriere

Status: Final

Visibility: Public

HISTORY

Document version #	Date	Remarks	Author
V1.0	22/12/2014	First draft	Anne Laverriere Bruno Farcy
V1.1	12/01/2015	Final after review	Anne Laverriere

TABLE OF CONTENTS

1. Executive Summary 5
2. Introduction 5
3. Use-cases: 2 modes to do the accounting 6
   3.1 Online consumption 6
   3.2 Offline consumption 6
4. Models 7
   4.1 Configuration model 7
   4.2 Calculation of the energy consumption 7
      4.2.1 Online consumption 7
      4.2.2 Offline consumption 8
   4.3 Process for determining the energy consumption 9
      4.3.1 Online consumption 9
      4.3.2 Offline consumption 10
   4.4 Prototyping 12
   4.5 APIs 12
      4.5.1 Configuration API 12
      4.5.2 API for calculating energy consumption Online 14
   4.6 Détermining an Online consumption 14
      4.6.1 Script to manage applications 14
   4.7 Determining the offline consumption 16
      4.7.1 Process of collect 16
      4.7.2 calculating energy consumption offline 18
   4.8 Implementation of the prototype 22
      4.8.1 Installation of offline collect: server OFF-LINE 22
      4.8.2 Installation of on-line local collect: server APPLI 22
      4.8.3 Tests 22
5. Conclusion 29
6. References 30
7. Abbreviations and Acronyms 30

TABLE OF FIGURES

- Figure 1: Modelized association between applications and HW to collect energy consumption data 7
- Figure 2: Online case: energy consumption process flow. 9
- Figure 3: Offline case: collect data flow 10
- Figure 4: Offline case: calculating process flow 11

1. Executive Summary

This document is part of the deliverables of Work Package 5.2 of the Cooperative H4H PERFCLOUD Project. This work package aims at improving the energetic efficiency of HPC cluster.

This deliverable presents the concepts and the results of an accounting solution to associate energy consumption data with a run of an application. The collect solution part was described in the previous deliverables of this task D5-2.1.

In the first place, it aims at describing two different concepts and models used to associate the collect mode for accounting and a running application. Indeed, we used and compared these two uses-cases (or two modes of accounting): “Online” or “Offline”.

The accounting results will show the granularity and the precision of the consolidated energy consumption calculated by these both ways.

2. Introduction

This paper aims to describe the development of an energy consumption information collection system by an application via the associated hardware resources.

Two methods are described to be adapted to different use-cases. An online method when we can directly put the accounting mechanism in the call of the application run, or an offline method if we had to do the accounting afterwards.

3. Use-cases: 2 modes to do the accounting

Two use cases are considered:

• "Online" consumption: the consumer information is automatically provided to each use of the application.

• "Offline" consumption: the consumer information is provided afterwards on request, for an application and for a given period.

3.1 Online consumption

The process of measuring the energy consumption is automatically triggered when starting the application and the consolidated value is provided upon application shutdown. This case requires:

- the availability of an application providing the energy consumption over time.
- the availability of APIs to start and stop the process of calculating the energy consumption, these APIs being used by any application.
- the establishment of a configuration system for associating a given application to the hardware resources.

3.2 Offline consumption

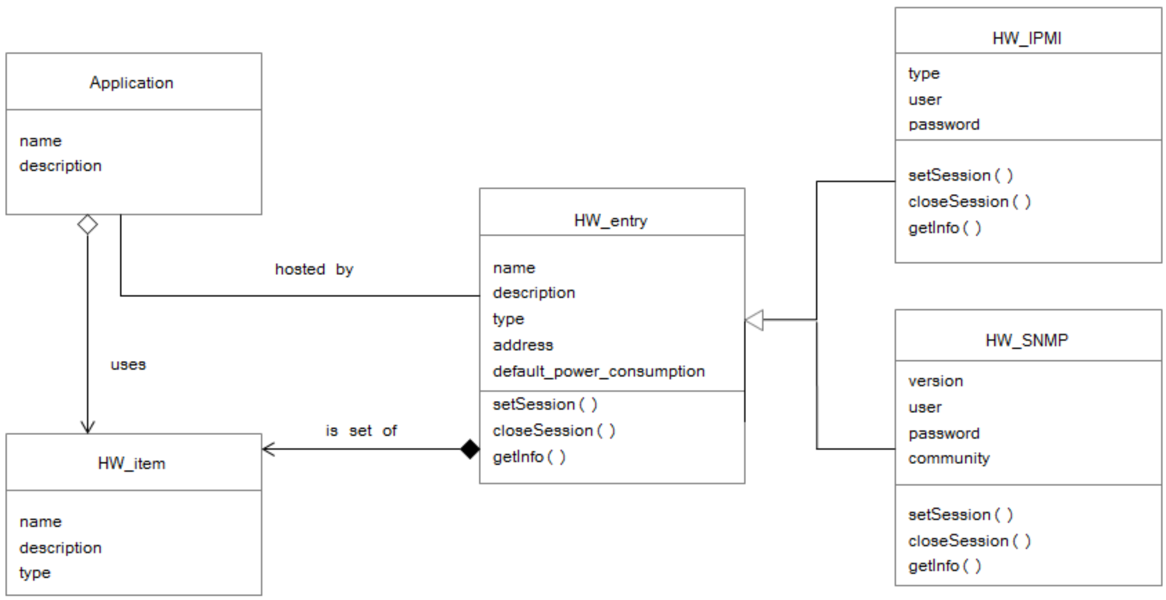
The energy consumption for an application can be requested afterwards, for a given period. This case requires:

- the establishment of a process of periodic collection and permanent indicator.
- defining rules for determining consumption over:
  - calculation rules between two collected values, depending on the type of indicator (consumption in Watt * h from a time t0 or direct value in Watts)
  - calculation rules to determine the assumed consumption data to the limits of the period if they do not exactly match the collection period
- the establishment of a configuration system for associating a given application to the hardware resources

4. Models

4.1 Configuration model

This following UML model proposes a way to associate an application to physical elements whose energy consumption can be measured:



• Figure 1: Modelized association between applications and HW to collect energy consumption data

An “Application” object is associated to a set of “HW_item” object, whose energy consumption data can be get via an “HW_entry” object, which can use different types of network protocol to provide data (IPMI or SNMP).

4.2 Calculation of the energy consumption

The energy consumption of an application is the sum of the consumption of all hardware components it uses.
The calculation of the consumption depends on how data collection is carried out and what kind of value, Consolidated Energy (Watt * h) or power (Watt), returns the hardware API.

4.2.1 Online consumption

This use case only supports consolidated energy values.
The start of the energy consolidation process being globally for all hardware components of an input, it is assumed that the mechanism is activated for any hardware component used by an application (with control from the application mechanism with possibly start if necessary)

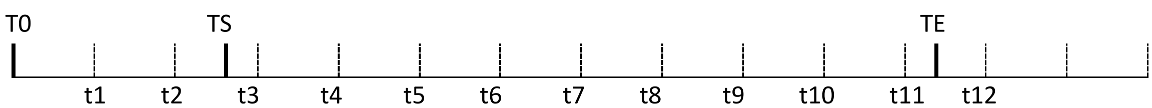
The application of consumer value is the difference between the value obtained at the end of application and that obtained at the beginning of application:

$$\mathrm{CONS\_WATTH} = \mathrm{val\_end} - \mathrm{val\_begin}$$

4.2.2 Offline consumption

In the case of offline use, the calculation is different depending on whether the values obtained consolidated energy or average power.
The diagram below shows the times at which:

- the consolidation process has started (T0)
- defining the time period over which consumption is to be estimated (TS, TE)
- the times at which the data is collected (tx).



4.2.2.1 Calculation of consumption from consolidated values (Watt * H)

In the case of consolidated values, consumption over is the difference between the value obtained at the period end and that obtained at beginning of period:

$$\mathrm{CONS\_TE} = \mathrm{val\_TE} - \mathrm{val\_TS}$$

When requested period is not precisely defined by a collection time, it must then estimate consumption border, taking as a correction factor, the ratio
(time_collect - time_period) / collection interval.

In the previous example, the total consumption of Watt during the period TS-TE can be estimated:

$$\mathrm{CONS\_WATTH} = (\mathrm{val\_t11} + (\mathrm{val\_t12} - \mathrm{val\_t11}) * (\mathrm{TE} - 11) / (\mathrm{t12} - \mathrm{t11})) - (\mathrm{Val\_t3} + (\mathrm{val\_t3} - \mathrm{val\_t2}) * (\mathrm{t3} - \mathrm{TS}) / (\mathrm{t3} - \mathrm{t2}))$$

4.2.2.2 Calculate consumption from direct values (Watt)

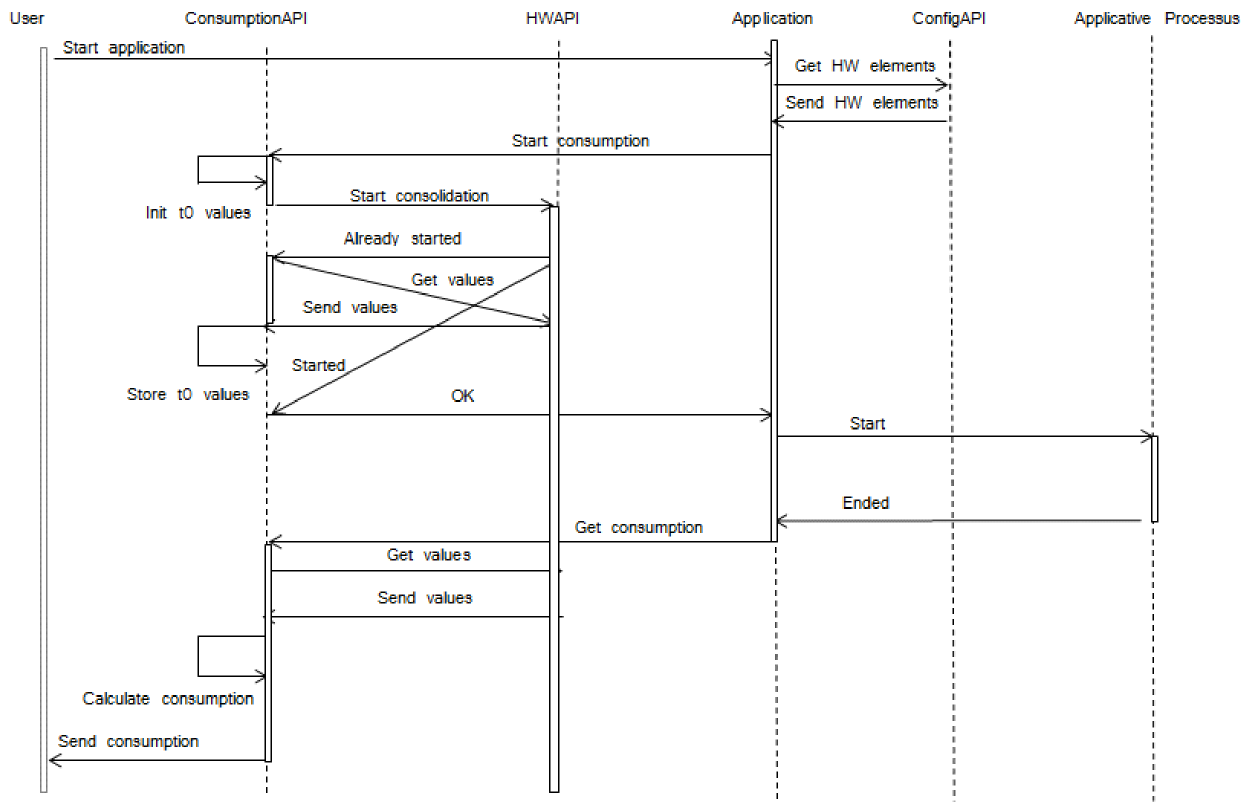
In the case of average power consumption for a given period is determined by the sum of consumption for each time interval, which is the product of power by time. For each time interval, we determine the average of the two powers obtained and the time being expressed in seconds, and we divide by 3600.

$$\mathrm{CONS\_WATTH} = \Sigma((\mathrm{ty} - \mathrm{tx}) * (\mathrm{x} + \mathrm{val} - \mathrm{val\_y}) / 2 * 3600)$$

4.3 Process for determining the energy consumption

4.3.1 Online consumption

The UML diagram below shows the exchanges between the elements involved in the process of determining the Online energy consumption.



• Figure 2: Online case: energy consumption process flow.

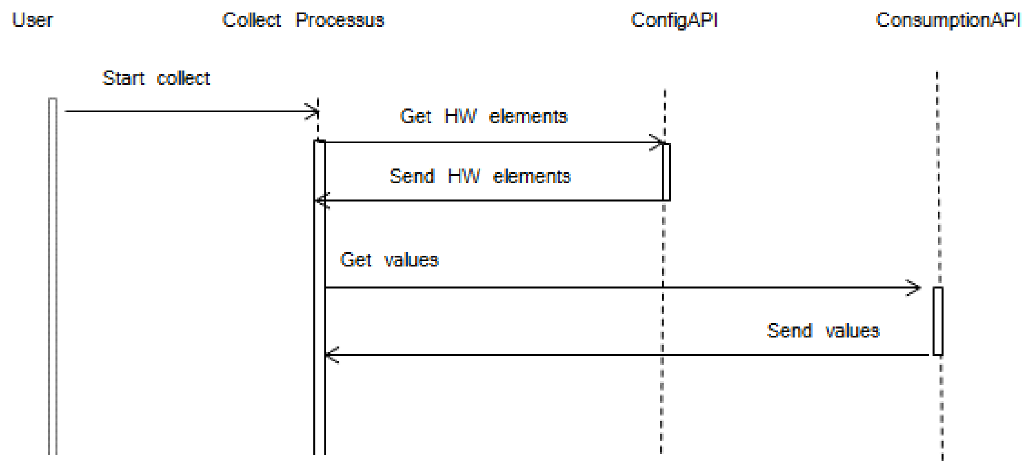
The Application element is an element required to manage an application process and consumer interface.

The ConsumptionAPI element represents an interface to enable an application to release and retrieve the consolidated energy consumption values.

The ConfigAPI element represents an interface to allow an application to determine what hardware resources it is associated.

4.3.2 Offline consumption

In the case of using offline consumption, it is assumed that collection process is continuously active to collect and store consumption data, at a sufficiently accurate frequency, to estimate consumption over time.
The diagram below shows the exchanges between the elements involved in the process of collecting energy consumption.



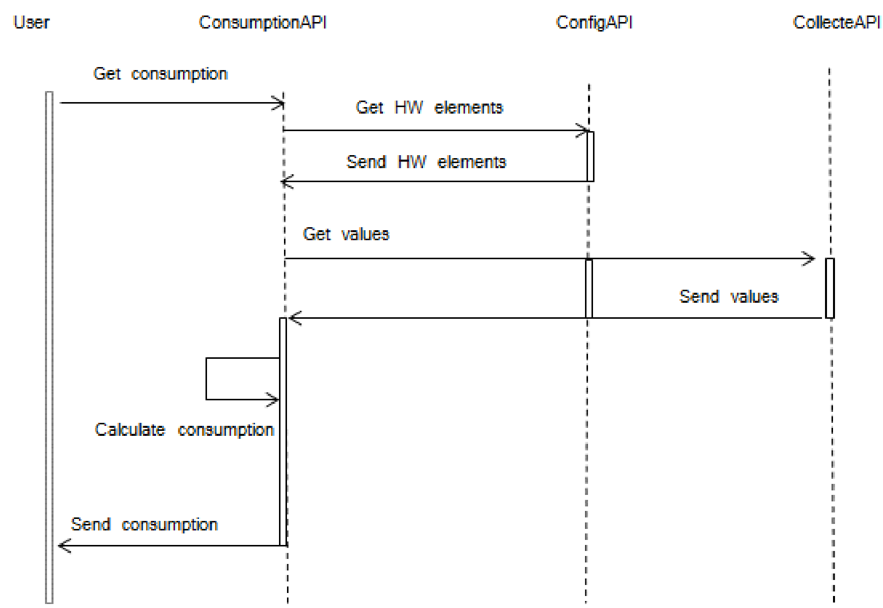
• Figure 3: Offline case: collect data flow

The Collect Processus element represents the mechanism set up to collect at regular intervals, energy consumption data.

The ConsumptionAPI element represents an interface to enable an application to release and retrieve the consolidated energy consumption values.

The ConfigAPI element represents an interface to allow an application to determine what hardware resources it is associated.

The diagram below shows the exchanges between the elements involved in the process of determining the offline energy consumption.



• Figure 4: Offline case: calculating process flow

The ConsumptionAPI element represents an interface to enable an application to release and retrieve the consolidated energy consumption values.

The ConfigAPI element represents an interface to allow an application to determine what hardware resources it is associated.

The CollectAPI element is an interface to allow an application to retrieve the values collected for a hardware item.

4.4 Prototyping

The proposed prototype aims at illustrating:

- Determining the Offline consumption of an application and include the following:
  - Configuration files of application combining the various hardware components and an API to retrieve the information,
  - A collection process for all configured hardware elements
  - A process to determine the consumption of an application over time.

- Determining the Online consumption of application and include the following:
  - Configuration files of application combining the various hardware components and an API to retrieve the information,
  - An API to access consolidated energy information, call the process manager application.

4.5 APIs

4.5.1 Configuration API

For the prototype, the configuration API allows reading the defined configuration files that need to be edited manually.

4.5.1.1 Configuration files

Whatever the prototype, two configuration files are defined:

- A file defining the characteristics of hardware components, named *HW_entries.cfg*.
- A file for applications involving a set of hardware elements, called *Applications.cfg*.

The files consist of a sequence of lines with the following format:

- [Ident] to indicate that begins a definition section of a hardware element or application.
- Attribute = value for characterizing a property of a hardware component or application.

Properties of a hardware element:

description	element description (optional)
type	type of protocol to access to indicators ex: IPMI, SNMP
address	IP address of the HW element
authentification	Authentication information to access to a remote interface (IPMI)

Sample of file HW_entries.cfg:

```
[bullx95]
description=prototype 1
type=IPMI
address=172.31.120.95
user=super
password=pass
default_power_consumption=400
cpu_list=CPU0_PVCCP,CPU0_PVSA,CPU1_PVCCP,CPU1_PVSA
ddr_list=DDR_AB,DDR_CD,DDR_EF,DDR_GH
```

Properties of an Application :

description	description of the application (optional)
hw_entry	name of the hardware component associated with the application This name must match a section defined in the configuration file of the hardware elements
item_list	names of hardware items associated with the application

Sample of file Applications.cfg:

```
[applisleep.sh]
description=BSM application
hw_entry=bullx95
item_list=CPU0_PVCCP,CPU0_PVSA,CPU1_PVCCP,CPU1_PVSA,DDR_AB,DDR_CD,DDR_EF,DDR_GH
[applihvt.sh]
description=hvt test
hw_entry=bullx95
item_list=CPU0_PVCCP,CPU0_PVSA,CPU1_PVCCP,CPU1_PVSA,DDR_AB,DDR_CD,DDR_EF,DDR_GH
[appliptumem.sh]
description=ptumem test
hw_entry=bullx95
item_list=CPU0_PVCCP,CPU0_PVSA,CPU1_PVCCP,CPU1_PVSA,DDR_AB,DDR_CD,DDR_EF,DDR_GH
```

4.5.1.2 API

The configuration API is defined in the getConfig.inc file that defines the following functions:

- *getAllHardwareIPMI*
  - ⇒ returns a set of rows that describes the IPMI hardware and items to monitor, with the following format: <hw_entry>; <address>; <user>; <password>; <item_list>. Used in particular by the collect scripts

- *getAppliHardwareInfo <appli_name>*
  ⇒ returns the HW information linked to an application with the format: <hw_entry>:<item_list>

- *getIPMIHardwareInfo <hw_entry>*
  ⇒ returns the IPMI parameters linked to a HW element with the format <hw_entry>;<address>;<user>;<password>

4.5.2 API for calculating energy consumption Online

The API calculation is defined in the getEnergy.inc file that defines the following functions:

- *getEnergyConsolidatedState <hw_entry>*
  ⇒ tests the status of the consolidation process for hardware element

- *getEnergyConsolidated <step> <appli_name> <mode> <consumption_dir>*
  ⇒ get consolidated energy for an application identified by its name.
  The step parameter can have the start or end value,
  The consumption_dir parameter specifies the directory where consumer information is stored (for later use by the getEnergyConsumption function)
  The mode parameter indicates whether the information is retrieved locally or remotely.

- *getEnergyConsumption <consumption_dir>*
  ⇒ calculation of energy consumption from data stored in the directory consumption_dir

4.6 Détermining an Online consumption

4.6.1 Script to manage applications

The launchAppli.sh script allows you to run an application and calculate the associated power consumption.

```
Usage:  launchAppli.sh -a <appli_name> -p <appli_parameters> -m <mode>
```

The mode parameter indicates whether the application is running on the hardware platform or remotely: according to the mode the IPMI queries will be executed in-band (via the OS driver) or out-off band (via IPMIoverLAN).

Exemple :

```
# /launchAppli.sh -a "applisleep.sh" -p "wait 60" -m local
start ./applisleep.sh, wait 60
```

```
./applisleep.sh ended
```

The script traces in file launchApplic.log the execution of applications, with their dates of beginning and end, as well as energy consumption.

Exemple:

```
--------------------------------------------------------------------------------
                     applihvt.sh -t melita_linpack -w 10800
                           on bullx95 (172.31.120.95)
--------------------------------------------------------------------------------
start the 12/15/14 at 16:45:06
end the 12/15/14 at 19:45:08
--------------------------------------------------------------------------------
Total energy consumption: 598.915
Total CPU consumption:    512.624
- CPU0            255.711
- CPU1            256.913
Total Memory consumption:  86.291
- DDR_EF          23.834
- DDR_CD          19.961
- DDR_GH          19.990
- DDR_AB          22.506
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
                             applisleep.sh Wait 600
                           on bullx95 (172.31.120.95)
--------------------------------------------------------------------------------
start the 12/15/14 at 19:45:08
end the 12/15/14 at 19:55:08
--------------------------------------------------------------------------------
Total energy consumption: 8.532
Total CPU consumption:    6.200
- CPU0            2.998
- CPU1            3.202
Total Memory consumption:  2.332
- DDR_EF          0.667
- DDR_CD          0.500
- DDR_GH          0.503
- DDR_AB          0.662
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
                   applihvt.sh -t melita_ubench_cpu -w 10800
                           on bullx95 (172.31.120.95)
--------------------------------------------------------------------------------
start the 12/15/14 at 19:55:08
end the 12/15/14 at 22:55:09
--------------------------------------------------------------------------------
Total energy consumption: 616.982
Total CPU consumption:    568.788
- CPU0            281.205
- CPU1            287.583
Total Memory consumption:  48.194
```

```
- DDR_EF          12.023
- DDR_CD          11.562
- DDR_GH          10.605
- DDR_AB          14.004
--------------------------------------------------------------------------------
```

Energy consumption is also stored in a specific output file of the application, identified by <*appli_name*> _ <*parameters*> _ <*date*> in the directory consumption. And for example applisleep.sh_wait-60_2014-12-17_15-10-18, the content is:

```
start the 12/17/14 at 15:10:18
end the 12/17/14 at 15:11:18
--------------------------------------------------------------------------------
Total energy consumption: 0.790
Total CPU consumption:    0.606
- CPU0            0.287
- CPU1            0.319
Total Memory consumption:  0.184
- DDR_EF          0.050
- DDR_CD          0.017
- DDR_GH          0.050
- DDR_AB          0.067
```

4.7 Determining the offline consumption

4.7.1 Process of collect

As described in deliverable D5-2.1.3, the collection process used for this prototype is *collectd*.

collectd is a daemon that collects statistics periodically and provides a mechanism to store values (RRD file, csv format).

The shell script called by collectd must perform the following steps:

- retrieves configuration information to list hardware items to collect.
- collectes at regular intervals, the consolidated usage information for each element.

Exemple: collectEnergyConsumption.sh

```
#!/bin/bash

. /home/al/POWER/api/getConfig.inc

hw=($(getAllHardwareIPMI))

IPMITOOL_CMD=/opt/BSMHW/bin/ipmitool
DEBUG_FILE=/tmp/collectenergy.debug
DEBUG=0
```

```
while sleep 10; do
     i=0
     while [ $i -lt  ${#hw[*]} ]
     do
    OIFS=$IFS
    IFS=','
    cmd_item_array=($(echo ${hw[$i]} | awk -v q="'" -F';' '{gsub(" ",":",$5);printf
"-H %s -U %s -P %s,%s,:%s: %s",$2,$3,$4,$1,$5,$6}'))
    base_cons=300
    IFS=$OIFS
    # execute cmd
    IPMITOOL_PARAMS="-I lan ${cmd_item_array[0]} bulloem powerstat energy"
    $IPMITOOL_CMD $IPMITOOL_PARAMS > /tmp/energy$$.out
    # parse output
    # process output based on the following output
    # BEGIN ----------------------------------------------------
    # Measurement state             : <state>
    # Stop state                    : <state>
    # Number of items               : <x>

    # Item    | Duration                                        | Consumption
(Watt*hour)
    # <item>  | <x>d <x>h <xx>mn <xx>s <xxx>ms <xxx>us <xxx>ns  | <x....x>.<xxx>
    # ................
    # <item>  | <x>d <x>h <xx>mn <xx>s <xxx>ms <xxx>us <xxx>ns  | <x....x>.<xxx>
    # END ----------------------------------------------------
    awk -v out=/tmp/collectEnergyValues.txt -v base_cons=$base_cons -v
name="${cmd_item_array[1]}/energy/power-" -v hwlist="${cmd_item_array[2]}" -F'|'
'BEGIN {tot=base_cons;process_item=0;running=1;}
      $1 ~ /Measurement state/ { if (!index($1,"Running")) {running=0; }}
      $1 ~ /Item.*/ { process_item=1;}
      $1 !~  /Item.*/ && $1 !~ /^$/ {
      if (process_item && running)  {
          gsub(/^[ ]+/,"",$1);
     gsub(/[ ]+$/,"",$1);
     if (match(hwlist,":"$1":")) {
            tot=tot+$3;
      printf "PUTVAL \"%s%s\" interval=10 N:%.3f\n",name,$1,$3;
          }
     }
      } END {
    if (!running) {exit 1;}
    }'  /tmp/energy$$.out
    rm -f /tmp/energy$$.out
    i=$(($i+1))
     done
done
```

4.7.2 calculating energy consumption offline

The determination of the energy consumption for a given application and a given period is carried out by a shell script to perform the following steps:

- Depending on the application (parameter), list the hardware elements whose values are to be collected.
- depending on the time, retrieving values near the period, for all the hardware elements associated or only for the application.
- make the calculation of consumption.

Exemple : script getEnergyConsumption.sh

```
#!/bin/bash

. /home/al/POWER/api/getConfig.inc

usage () {
    echo "getEnergyConsumption.sh -a <application_ident> -s <start-date MM/DD/YYYY
HH:MM:SS> -e <end-date MM/DD/YYYY HH:MM:SS>"
}

export LC_NUMERIC=en_US.UTF-8
while getopts a:s:e:r: option
do
        case $option in
        a) APPLI_NAME=${OPTARG};continue;;
        s) START_DATE=${OPTARG};continue;;
        e) END_DATE=${OPTARG};continue;;
        r) RRD_BASE=${OPTARG};continue;;
        ?) usage; exit 0;;
        esac
done

if test -z "$APPLI_NAME"; then
    usage
    echo "The ident of the application is not set"
    exit 2
fi


if test -z "$START_DATE"; then
   usage
    echo "The start date is not set"
    exit 2
fi

if test -z "$END_DATE"; then
   usage
```

```
    echo "The end date is not set"
    exit 2
fi

# ctrl date
sts=`date -d "$START_DATE" +%s`
ets=`date -d "$END_DATE" +%s`
if [ $sts -gt $ets ]; then
    echo "Invalid date: end date prior begin date"
    exit 2
fi

if test -z "$RRD_BASE"; then
  COLLECT_DB_DIR=/var/lib/collectd/rrd
else
  COLLECT_DB_DIR=/var/lib/collectd/$RRD_BASE
fi
RRDTOOL_CMD=/usr/bin/rrdtool
RRD_INDIC=energy/power
DEBUG_FILE=/tmp/getEnergyConsumption.debug
DEBUG=0
current_val=0

[ $DEBUG ] && echo "------------------------------------------------------------" >
$DEBUG_FILE

# APPLI_NAME=bsm
# START_DATE="01/21/2014 09:59:00"
# END_DATE="01/15/2014 10:35:00"

function listHwItem () {

    [ $DEBUG ] && echo "... in listHwItem" >> $DEBUG_FILE
    OIFS=$IFS
    IFS=','
    hw_info=($(getAppliHardwareInfo $APPLI_NAME | sed 's/:/,/'))
    IFS=$OIFS
}

function buildRddFiles () {
  [ $DEBUG ] && echo "... in buildRddFiles" >> $DEBUG_FILE
  [ $DEBUG ] &&  echo "nb elemt = ${#hw_info[*]}" >> $DEBUG_FILE
    i=1
    j=0
    HW_ENTRY=${hw_info[0]}
    while  [ $i -lt  ${#hw_info[*]} ] ; do
       RRD_FILE[$j]=$COLLECT_DB_DIR/$HW_ENTRY/${RRD_INDIC}-${hw_info[$i]}.rrd
       ITEM_NAME[$j]=${hw_info[$i]}
       [ $DEBUG ] &&  echo "ITEM ${hw_info[$i]} =>
$COLLECT_DB_DIR/$HW_ENTRY/${RRD_INDIC}-${hw_info[$i]}.rrd" >> $DEBUG_FILE
       i=$(($i+1))
       j=$(($j+1))
    done
```

```
}

function getRrdValue () {
    rrd_file=$1
    ts=$2
    [ $DEBUG ] && echo "search value for $ts"  >> $DEBUG_FILE
    current_ts=`date +%s`
    # retrieve values around ts
    delta_ts=$(($current_ts - $ts))
    [ $DEBUG ] && echo "current_ts = $current_ts"  >> $DEBUG_FILE
    [ $DEBUG ] && echo "delta_ts = $delta_ts"  >> $DEBUG_FILE
    # 1d=86400s 2d=172800 4d=345600 8d=691200
    if [ $delta_ts -gt 691200 ]; then
       delta_a=3000
    elif [ $delta_ts -gt 345600 ]; then
       delta_a=1500
    elif [ $delta_ts -gt 172800 ]; then
       delta_a=800
    elif [ $delta_ts -gt 86400 ]; then
       delta_a=300
    else
       delta_a=20
    fi
    [ $DEBUG ] && echo "delta_a = $delta_a"  >> $DEBUG_FILE
    delta_e=$(($delta_a * 2))
    ats=$(($ts - $delta_a))
    ets="s+${delta_e}s"
    [ $DEBUG ] && echo "ats=$ats ets=$ets" >> $DEBUG_FILE
    [ $DEBUG ] &&  echo "execute $RRDTOOL_CMD fetch --start $ats --end $ets $rrd_file
AVERAGE | grep \"^[0-9]\" " >> $DEBUG_FILE
    $RRDTOOL_CMD fetch --start $ats --end $ets $rrd_file AVERAGE | grep "^[0-9]" >
/tmp/result$$
    [ $DEBUG ] &&  cat /tmp/result$$ >> $DEBUG_FILE
    awk  -v ts=$ts -v debug=$DEBUG -v debugfile=$DEBUG_FILE -F':' 'BEGIN {inf_ts=0;
sup_ts=0; inf_val=0; sup_val=0; final_val=0}
       $1 < ts {inf_ts=$1;inf_val=sprintf("%.3f",$2)}
       $1 > ts {if (sup_val == 0) {sup_ts=$1;sup_val=sprintf("%.3f",$2)}}
       $1 == ts {gsub(/,/,".",$2);final_val=sprintf("%.3f",$2)}
       END { if (final_val == 0 ) {
                if (debug) {printf("inf: ts=%s val=%.3f \n",inf_ts,inf_val) >>
debugfile}
                if (debug) {printf("sup: ts=%s val=%.3f \n",sup_ts,sup_val) >>
debugfile}
                final_val = inf_val + (sup_val -inf_val) * (sup_ts - ts) / (sup_ts -
inf_ts);
             }
             if (debug) {printf("final: ts=%s val=%.3f \n",ts,final_val)>> debugfile}
             print(final_val)}' /tmp/result$$
}

function calculateConsumption () {
  [ $DEBUG ] && echo "... in calculateConsumption" >> $DEBUG_FILE
    i=0
```

```
    totalcons=0
    while [ $i -lt ${#ITEM_NAME[*]} ]; do
       [ $DEBUG ] &&  echo ">>> search consumption for ${ITEM_NAME[$i]} ...." >>
$DEBUG_FILE
       [ $DEBUG ] &&  echo "--- begin value ($START_DATE/$sts)" >> $DEBUG_FILE
       stval=`getRrdValue ${RRD_FILE[$i]} $sts`
       [ $DEBUG ] &&  echo "--- end value ($END_DATE/$ets)" >> $DEBUG_FILE
       endval=`getRrdValue ${RRD_FILE[$i]} $ets`
        [ $DEBUG ] &&  echo "stval = $stval, endval=$endval" >> $DEBUG_FILE
       itemcons=`echo "scale=3;$endval - $stval" | bc`
       echo "$APPLI_NAME ($START_DATE -> $END_DATE) : ${ITEM_NAME[$i]} (cons) =
$itemcons"  | tee -a $DEBUG_FILE
       [ $DEBUG ] &&  echo "=> itemcons = $itemcons" >> $DEBUG_FILE
       totalcons=`echo "scale=3;$totalcons + $itemcons" | bc`
       [ $DEBUG ] &&  echo "=> totalcons = $totalcons" >> $DEBUG_FILE
       i=$(($i+1))
    done
    echo "$APPLI_NAME ($START_DATE -> $END_DATE) : total (cons) = $totalcons" | tee -a
$DEBUG_FILE
       rm -f /tmp/result$$
}

listHwItem
buildRddFiles
calculateConsumption
```

```
# ./getEnergyConsumption.sh -a applisleep.sh -s "12/17/14 15:10:18" -e "12/17/14
15:11:18"
applisleep.sh consumption
--------------------------------------------------------------------------------
start the 12/17/14 at 15:10:18
end the 12/17/14 at 15:11:18
--------------------------------------------------------------------------------
Total energy consumption: 0.802
Total CPU consumption:    0.617
- CPU0            0.294
- CPU1            0.323
Total Memory consumption:  0.185
- DDR_CD          0.017
- DDR_EF          0.050
- DDR_GH          0.050
- DDR_AB          0.068
```

In this calculation, we find a value very similar to that obtained with the Online method.

The prototype could be improved with an collect API over rrdtool, particularly in terms of the file naming rule RRA (accessed by the script collection and the calculation script).

4.8 Implementation of the prototype

4.8.1 Installation of offline collect: server OFF-LINE

The scripts needed to collect information on the OFF-LINE server are installed in the directory: /home /perfcloud:

- **config** subdirectory contains *HW_entries.cfg* and *Applications.cfg* files that describe the hardware platforms and applications used in the prototype.
- *collectEnergyConsumption.sh* script is installed in the *collectd-script* directory and will be called by collectd.
- the **api** subdirectory contains files describing API s configuration and consumption.
- the **offline-script** subdirectory contains the file *getEnergyConsumption.sh*.

4.8.2 Installation of on-line local collect: server APPLI

Scripts and configuration files are installed in the server running the applications in the /home/perfcloud:

- **config** subdirectory contains *HW_entries.cfg* and *Applications.cfg* files that describe the hardware platforms and applications used in the prototype (identical to those installed on the server OFF-LINE)
- the **api** subdirectory contains files describing API s configuration and consumption.
- Script to run applications with calculation of on-line consumption (*launchAppli.sh*) is installed in /home/perfcloud.

The tested applications are:

- applisleep.sh <p1> <p2>:
    - This application runs a sleep of <p2> seconds. The <p1> is the message displayed by the application.
- applihvt.sh -t <testname> -w <ElapsedTime>
    - This application runs a hvt test (load test identified by testname) over a period ElapsedTime.

4.8.3 Tests

4.8.3.1 Campaign tests 1

<u>Online Consumption :</u>

Several applications have been executed sequentially over a period of 24h.

The table below summarizes the tests run, duration and consumption of information recorded in the online consumer files.

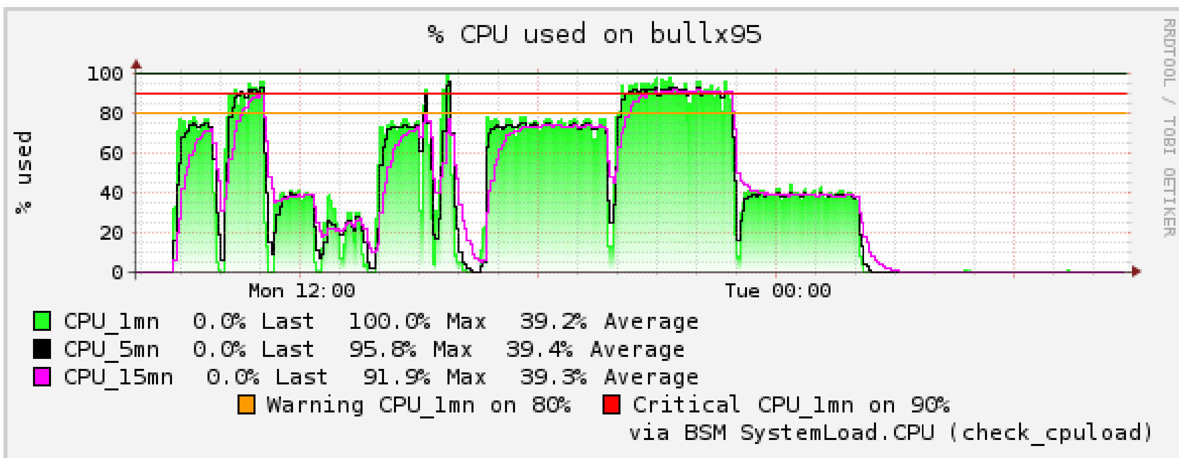
Application	**Time**	**Online Consumption**
hvt (melita_linpack)	16:45:06 19:45:08	Total energy consumption: 598.915 Total CPU consumption: 512.624 - CPU0 255.711 - CPU1 256.913 Total Memory consumption: 86.291 - DDR_EF 23.834 - DDR_CD 19.961 - DDR_GH 19.990 - DDR_AB 22.506
hvt (melita_ubench_cpu)	19:55:08 22:55:09	Total energy consumption: 616.982 Total CPU consumption: 568.788 - CPU0 281.205 - CPU1 287.583 Total Memory consumption: 48.194 - DDR_EF 12.023 - DDR_CD 11.562 - DDR_GH 10.605 - DDR_AB 14.004
hvt(melita_memtester)	23:05:10 02:05:12	Total energy consumption: 503.872 Total CPU consumption: 382.664 - CPU0 191.008 - CPU1 191.656 Total Memory consumption: 121.208 - DDR_EF 32.481 - DDR_CD 28.960 - DDR_GH 28.614 - DDR_AB 31.153
applisleep	02:05:12 05:05:12	Total energy consumption: 152.565 Total CPU consumption: 110.218 - CPU0 53.046 - CPU1 57.172 Total Memory consumption: 42.347 - DDR_EF 12.012 - DDR_CD 9.008 - DDR_GH 9.957 - DDR_AB 11.370

The following graphs provide an overview of CPU and memory consumption during test execution.

Host: bullx95 **Service:** SystemLoad.CPU

25 Hours 15.12.14 7:48 - 16.12.14 8:48

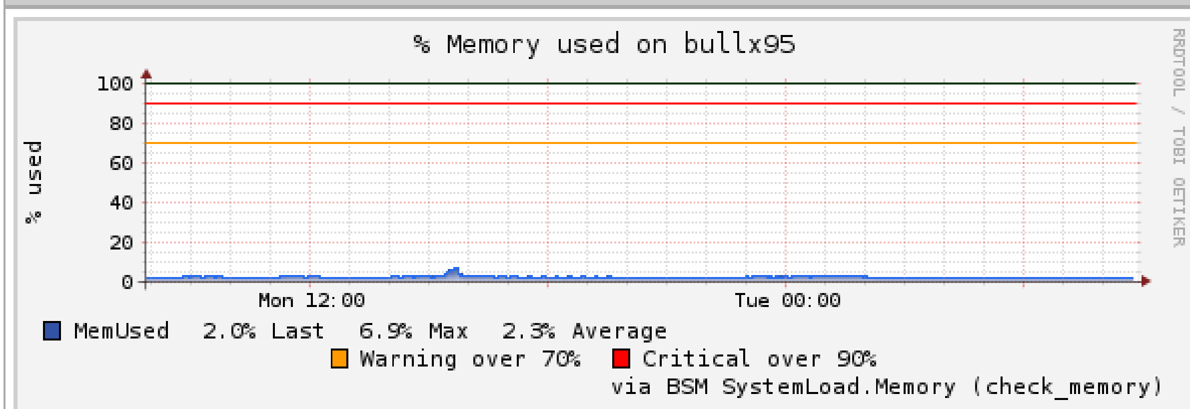
Datasource: CPU_1mn, CPU_5mn & CPU_15mn



Host: bullx95 **Service:** SystemLoad.Memory

25 Hours 15.12.14 7:49 - 16.12.14 8:49

Datasource: MemUsed



Comparison between consumption on-line/off-line

The following tables compare the on-line and off-line consumption.

Application hvt (melita_linpack) during 3h

on-line consumption (watt*hours)	off-line consumption (watt*hours)
Total energy consumption: 598.915	Total energy consumption: 598.747
Total CPU consumption: 512.624	Total CPU consumption: 512.480
- CPU0 255.711	- CPU0 255.640
- CPU1 256.913	- CPU1 256.841
Total Memory consumption: 86.291	Total Memory consumption: 86.266
- DDR_CD 19.961	- DDR_CD 19.955
- DDR_EF 23.834	- DDR_EF 23.827
- DDR_GH 19.990	- DDR_GH 19.985
- DDR_AB 22.506	- DDR_AB 22.500

Application hvt (melita_ubunch_cpu) during 3h

on-line consumption (watt*hours)	off-line consumption (watt*hours)
Total energy consumption: 616.982	Total energy consumption: 616.507
Total CPU consumption: 568.788	Total CPU consumption: 568.330
- CPU0 281.205	- CPU0 280.977
- CPU1 287.583	- CPU1 287.353
Total Memory consumption: 48.194	Total Memory consumption: 48.178
- DDR_CD 11.562	- DDR_CD 11.558
- DDR_EF 12.023	- DDR_EF 12.019
- DDR_GH 10.605	- DDR_GH 10.600
- DDR_AB 14.004	- DDR_AB 14.000

Application hvt (melita_memtester) during 3h

on-line consumption (watt*hours)	**off-line consumption (watt*hours)**
Total energy consumption: 503.872 Total CPU consumption: 382.664 - CPU0 191.008 - CPU1 191.656 Total Memory consumption: 121.208 - DDR_CD 28.960 - DDR_EF 32.481 - DDR_GH 28.614 - DDR_AB 31.153	Total energy consumption: 504.059 Total CPU consumption: 382.805 - CPU0 191.076 - CPU1 191.729 Total Memory consumption: 121.254 - DDR_CD 28.971 - DDR_EF 32.493 - DDR_GH 28.625 - DDR_AB 31.165

Application appisleep sur 3h

on-line consumption (watt*hours)	**off-line consumption (watt*hours)**
Total energy consumption: 152.565 Total CPU consumption: 110.218 - CPU0 53.046 - CPU1 57.172 Total Memory consumption: 42.347 - DDR_CD 9.008 - DDR_EF 12.012 - DDR_GH 9.957 - DDR_AB 11.370	Total energy consumption: 152.471 Total CPU consumption: 110.144 - CPU0 53.011 - CPU1 57.133 Total Memory consumption: 42.326 - DDR_CD 9.002 - DDR_EF 12.007 - DDR_GH 9.953 - DDR_AB 11.364

We notice the very similar results with both accounting methods. This confirms the accuracy of the consolidated data provided at the HW level, and the two proposed methods can be used to realize this kind of accounting.
As expected, we can noticed the big part of energy consumption used by CPUs, and also, the well-balanced energy consumption at the DDRAM risers level.

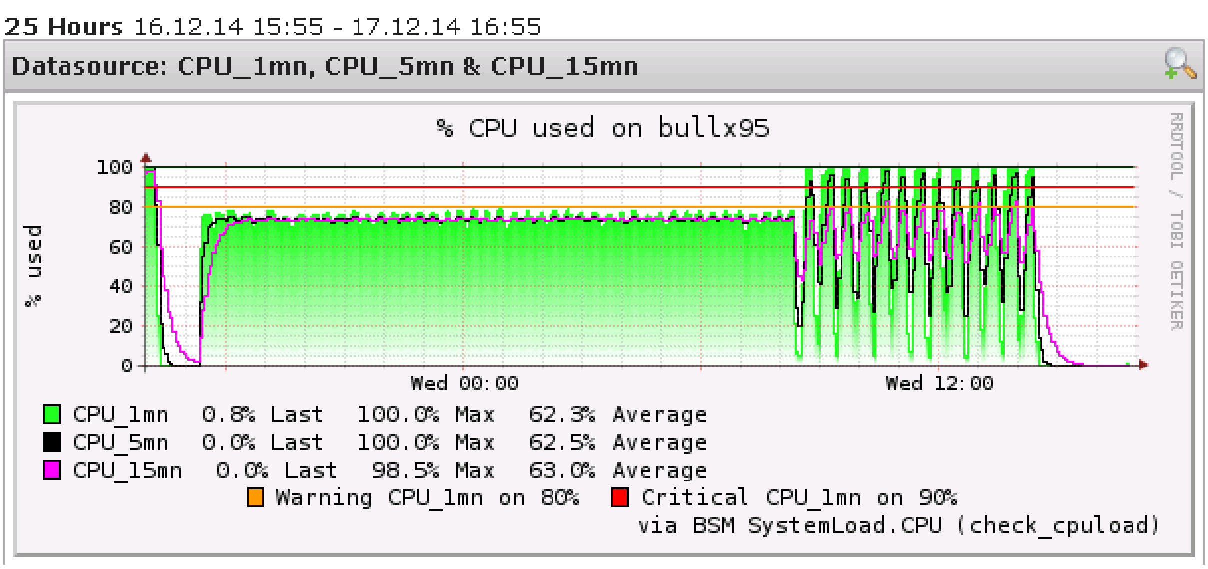
4.8.3.2 Campaign tests 2

<u>Online consumption</u>

The table below summarizes the tests run, duration and consumption of information recorded in the online consumer files.

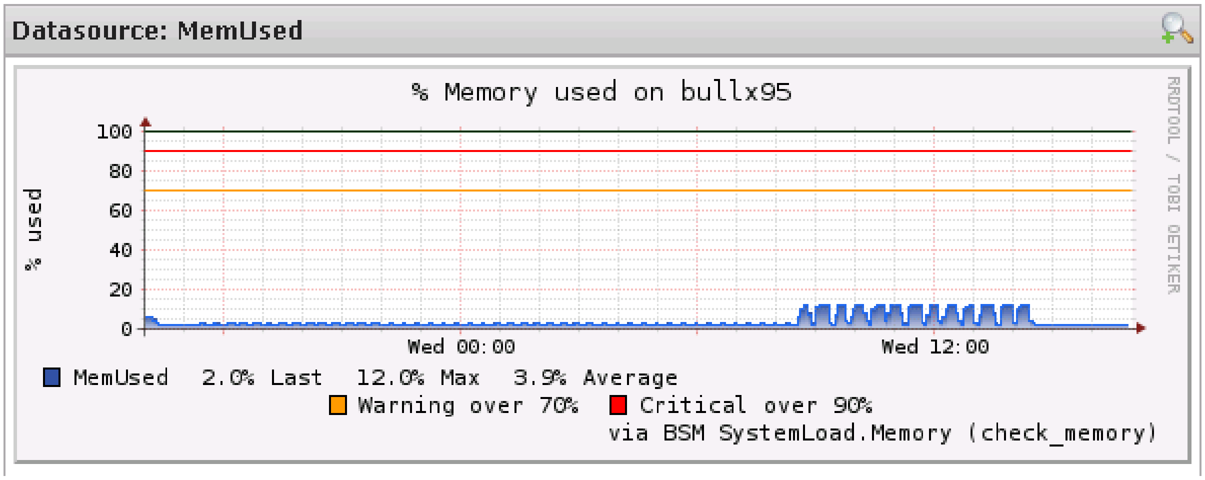
Application	Time	on-line consumption (watt*hours)
hvt (imts_memmiss)	08:26:26 14:26:26	Total energy consumption: 820.612 Total CPU consumption: 712.798 - CPU0 331.270 - CPU1 381.528 Total Memory consumption: 107.814 - DDR_EF 38.882 - DDR_CD 12.087 - DDR_GH 16.721 - DDR_AB 40.124

The graphs below provide an overview of CPU and memory consumption during test execution.



Host: bullx95 **Service:** SystemLoad.Memory

25 Hours 16.12.14 15:57 - 17.12.14 16:57



<u>Comparison between on-line/off-line consumption:</u>

The table below compares the on-line and off-line consumption.

Application hvt (immts_memmiss) during 6h

on-line consumption (watt*hours)	**off-line consumption (watt*hours)**
Total energy consumption: 820.612	Total energy consumption: 820.513
Total CPU consumption: 712.798	Total CPU consumption: 712.707
- CPU0 331.270	- CPU0 331.230
- CPU1 381.528	- CPU1 381.476
Total Memory consumption: 107.814	Total Memory consumption: 107.806
- DDR_EF 38.882	- DDR_EF 38.878
- DDR_CD 12.087	- DDR_CD 12.087
- DDR_GH 16.721	- DDR_GH 16.720
- DDR_AB 40.124	- DDR_AB 40.121

Once again, we can appreciate the very similar results between the two modes of accounting. As expected, we can also noticed the big part of energy consumption used by CPUs, and more surprising, the asymmetric energy consumption at the DRAM risers level (more concentrated on the first 4 DDRAMs) specific to this kind of bench.

5. Conclusion

This deliverable shows the description of a proof of concept of an accounting solution that is able to associate the run of an application software and the energy consumption of the resources that this application has used.
Two methods (online and offline) were proposed and both seems appropriated to provide a detailed accounting of energy consumption. By these ways, we can have an accounting method adapted to different use-cases and may be independent with the collect engine. Tonly need is in fact the granularity of the collected data and the data aggregation at the low level to provide the consolidated energy consumption (watt*hours).
So, the results of these first tests not only demonstrate the ability to do this accounting but also conclude and materialize all the study done in the task 5-2.1 about models, concepts and specification to create a software accounting solution in order to associate energy consumption data of HW items and an application run.

6. References

The three previous deliverables D5-2.1.1: “Energy Consumption Model for Accounting”, D5-2.1.2: “First version of software to collect energy data” and D5-2.1.3: “Final version of software to collect energy data”

7. Abbreviations and Acronyms

collectd: collectd is a daemon which collects system performance statistics periodically and provides mechanisms to store the values in a variety of ways, for example in RRD files..

IPMI: The **Intelligent Platform Management Interface** (IPMI) is a standardized computer system interface used by system administrators for out-of-band management of computer systems and monitoring of their operation.

RRDtool: RRDtool (acronym for **round-robin database tool**) aims to handle time-series data like network bandwidth, temperatures, CPU load, etc. The data are stored in a round-robin database (circular buffer), thus the system storage footprint remains constant over time.

SNMP: **Simple Network Management Protocol** (SNMP) is an "Internet-standard protocol for managing devices on IP networks". Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks and more.

UML: The **Unified Modeling Language** (UML) is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system