Shellscripts and Commands

practical development method from large-scale enterprise system to small tools development

BSD Consulting, Inc. (Tokyo) Director
ONGS Inc. CEO / FreeBSD Project

Daichi GOTO

introduction

- Daichi GOTO / 後藤大地 *1980〜*
- BSD Consulting, Inc. Director / ONGS Inc. CEO
- FreeBSD Project
- Enterprise system design / development / management and maintenance, web server design / development / management and maintenance, IT-related news / articles / magazine and books writing, IT-related seminar, IT-related consulting, etc

summary

- xRAD (Extremely Rapid Application Development method; 超高速開発手法)
- one of the xRADs for *BSD systems
- Overview: Unicage software development method
- Drill-down: Unicage software development method hands-on

xRAD

xRAD

- xRAD: Extremely Rapid Application Development Method
- Main members: Business managers of software vendors and their clients
- Objectives: to achieve high productivity, high flexibility and high quality of enterprise system
- Background: lower labor productivity relative to GDP in Japan, the situation is quite desperate in software industry
- Universal Shell Programming Laboratory (USP) is one of the original members
- USP's xRAD method is "Unicage software development method" (Unicage)

Unicage software development method

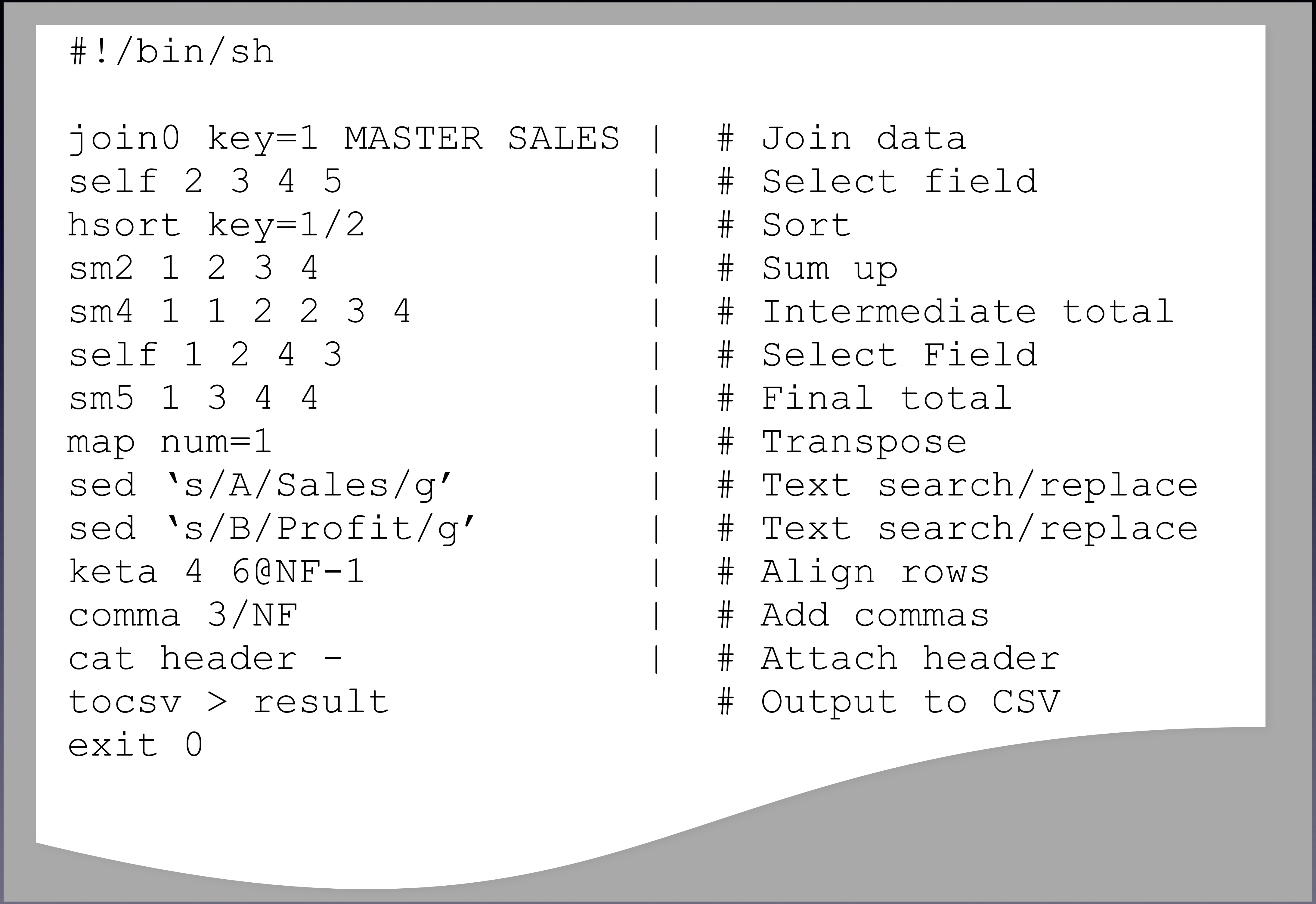
Unicage software development method

- Unicage: one of the xRAD
- Applicable scope: how to communicate with clients, requirement definition, how to organize team, deploying servers, files and directories structure, how to use commands and shellscripts, coding manners, etc.

Unicage main programming aspect

- No middleware
- No relational databases
- No if / while / for / function() / var=
- Yes space separated test file
- Yes simple commands
- Yes pipelines
- Yes redirects
- Yes files instead of variables
- Flows from up to down

Unicage: sales report example



```
#!/bin/sh

join0 key=1 MASTER SALES |  # Join data
self 2 3 4 5             |  # Select field
hsort key=1/2            |  # Sort
sm2 1 2 3 4              |  # Sum up
sm4 1 1 2 2 3 4          |  # Intermediate total
self 1 2 4 3             |  # Select Field
sm5 1 3 4 4              |  # Final total
map num=1                |  # Transpose
sed ‘s/A/Sales/g’        |  # Text search/replace
sed ‘s/B/Profit/g’       |  # Text search/replace
keta 4 6@NF-1            |  # Align rows
comma 3/NF               |  # Add commas
cat header -             |  # Attach header
tocsv > result              # Output to CSV
exit 0
```

From the “Unicage Development Method Technical Overview”, 2013 USP Lab.

usp Tukubai commands sample

Database Commands

join0,1,2:	Table join
gyo:	Count matching records
getfirst:	Get first matching row
getlast:	Get last matching row
retu:	Count columns

I/O Commands

cgi-name:	Read data from CGI-POST
mime-read:	Read MIME encoded data

Arithmetic Functions

plus:	Addition
divsen:	Divide by 1000
sm2:	Sum a field
marume:	Round a number
ratio:	Find a ratio
plus:	Sum all fields in a record

Formatting Commands

comma:	Add commas to number
mojihame:	Merge data into template
tcat:	Vertically concatenate
ycat:	Horizontally concatenate
map:	Transpose rows/columns
yobi	Get day of week
up3:	Merge files on key field

From the "Unicage Development Method Technical Overview", 2013 USP Lab.

Unix commands, Tukubai and client's custom cummands

Only 100 Common Commands

cp	Copy	FreeBSD Commands
find	File search	
sort	Sort	
awk	Perform operations on items	
:		
:		
join0	Data matching	usp Tukubai Commands
sm2	Sum up	
waku	Add a border	
ulock	Lock control	
:		
bdate	Date management	Custom Commands

Open usp Tukubai

- USP Labs is releasing open source licensed version of usp Tukubai “Open usp Tukubai” written in Python
- I imported it to FreeBSD devel/open-usp-tukubai
- http://uec.usp-lab.com/ helps you

Unicage: CGI example

```
#!/bin/sh

dd bs=$CONTENT_LENGTH | cgi-name > name   # Get information from web server
case “$(nameread MODE name)” in           # Branch based on processing mode
SEARCH)                                   # [Search]
    if ulock -r MST.LK; then              # Shared Lock
        nameread KEY name     |           # Get search key
        join0 key=1 - MST     |           # Search for master data
        mojihame -lLABEL html             # Export to HTML
    fi ;;
UPDATE)                                   # [Update]
    if ulock -w MST.LK; then              # Exclusive lock
        nameread -el “KEY|VAL” name > TRN.123 # Get key and value
        upl key=1 MST TRN.123 > MST.123   # Create update master
        ln -s MST.123 MST                 # Allow access with same name
        cat next_html                     # Output to next screen
    fi ;;
esac
exit 0
```

From the “Unicage Development Method Technical Overview”, 2013 USP Lab.

Master and Transaction

- Master and Transaction: space separated text file
- Master: contains primary key column, must be sorted, not updated at short intervals.
  ex., shop master, item master, customers master
- Transaction: primary key columns and sort is not necessary, updated at short intervals
  ex., sales data and log data

```
% head -5 *
==> FOOD_MASTER <==
000001 Vegetable
000002 Fish
000003 Beverage

==> SALES <==
000001 20140516130102 000001 235 3
000001 20140516130103 000002 923 4
000001 20140516130103 000003 524 1
000001 20140516130105 000001 625 1
000001 20140516130106 000002 223 3

==> STORE_MASTER <==
000001 Tokyo
000002 Ottawa
000003 Sunnyvale
000004 Cambridge
000005 Sofia
%
```

Files and Directories Structure

- LV1, LV2, LV3, LV4, LV5 : data directories
  no overwrite files, no remove, only create or append into files
- LV1: raw data file (POS-terminal data)
- LV2: SSV data files from LV1
- LV3: normalized SSV date files from LV2
- LV4: data files apps needed outside LV3
- LV5: data files apps generated
- Data flow: LV1→LV2→LV3→LV4→LV5

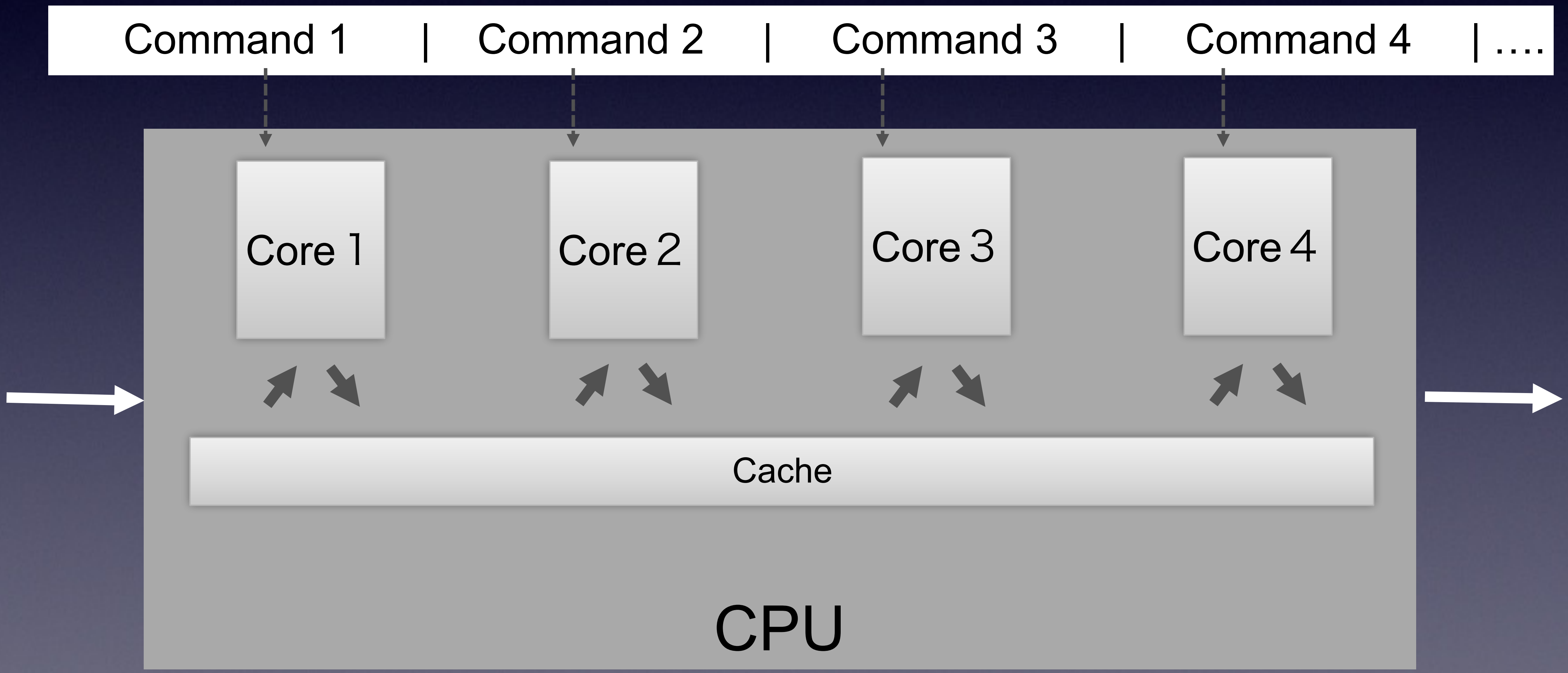
```
% tree
.
├── LV1
│   └── sales_20140516_01012123
├── LV2
│   ├── SALES_20140516_13.SSV
│   └── SALES_20140516_14.SSV
├── LV3
│   ├── FOOD_MASTER
│   ├── SALES
│   └── STORE_MASTER
├── LV4
│   ├── SALES.TXT
│   └── SALES_REPORT.HTML
├── LV5
│   ├── SALES_REPORT_20140516_13.TXT
│   └── SALES_REPORT_20140516_14.TXT
└── SHL
    ├── L2MAKE.SH
    └── MAKE_SALES_REPORT.SH

6 directories, 12 files
%
```

ACID of Unicage
not overwrite and original and duplicate H/W architecture

- RDMS engineers want to know ACID of Unicage
- Unicage has 2 rules to achieve ACID
- 1st: no overwrite data files
- 2nd: original and duplicate H/W architecture to compare the results of those two PCs assures reliability
- Original and 2 duplicate H/Ws for more reliability

Performance and Scalability multicore / many-core

Pipeline Processing : easy way to use multicore



From the "Unicage Development Method Technical Overview", 2013 USP Lab.

Coding manners

Shell Scripts are extremely flexible so we must pay close attention to proper style when using Unicage.

- Script header style
- Comment style
- Variable and file naming rules
- Rules for naming temporary files
- One command per line
- Transfer data using files (not environment variables)
- Include processing is forbidden
- File layout style
- Execution log style
- Rules for naming files
- Output execution start and end times
- Generate a semaphore file
- Keep it short
- Separate script and data in complex IF statements
- Delete garbage files
- Don’t create versions (but make backups)
- Multi-level calls prohibited
- Overwrite the copyright
- Understand the size of the processing file

From the “Unicage Development Method Technical Overview”, 2013 USP Lab.

Unicage Development Method Documentation

1. Very Little Documentation is Required for Development
    - Configuration of data and programs is fixed, so only basic documentation is necessary.
    - Required documents are as follows:
        - Application I/O API specifications
        - Dimensions of source data
2. Documentation for Understanding the System is Practical
    - "System Purpose", "Business Flow", "Manuals" are needed.
    - Most information needed to understand the system can be obtained by looking at the system operation itself (examples below)
        - Data configuration and relationships
        - Application configuration and relationships
        - Batch schedule
        - Detailed specifications (written in the shell scripts)

From the "Unicage Development Method Technical Overview", 2013 USP Lab.

Unicage software development method

graphic explanation

Development Flow

70% Important
for Business

Design Framework for Business

Can be Decided Early

Short Timeframe

Initial Development

Obtain Real Data
Develop, Test,
Release

Can be designed only with the original data and output format design (Design Workload is Light)

Few Man-Hours

30% Important
for Business

Design Useful Features

Based on Working Application so

Short Timeframe

Second Pass (Finishing)

Develop, Test,

Short Program
+
No Dependencies
↓
Feature Extension is Easy

Few Man-Hours

Short Program + User Focus + 2-Steps = Low Cost Fast Response

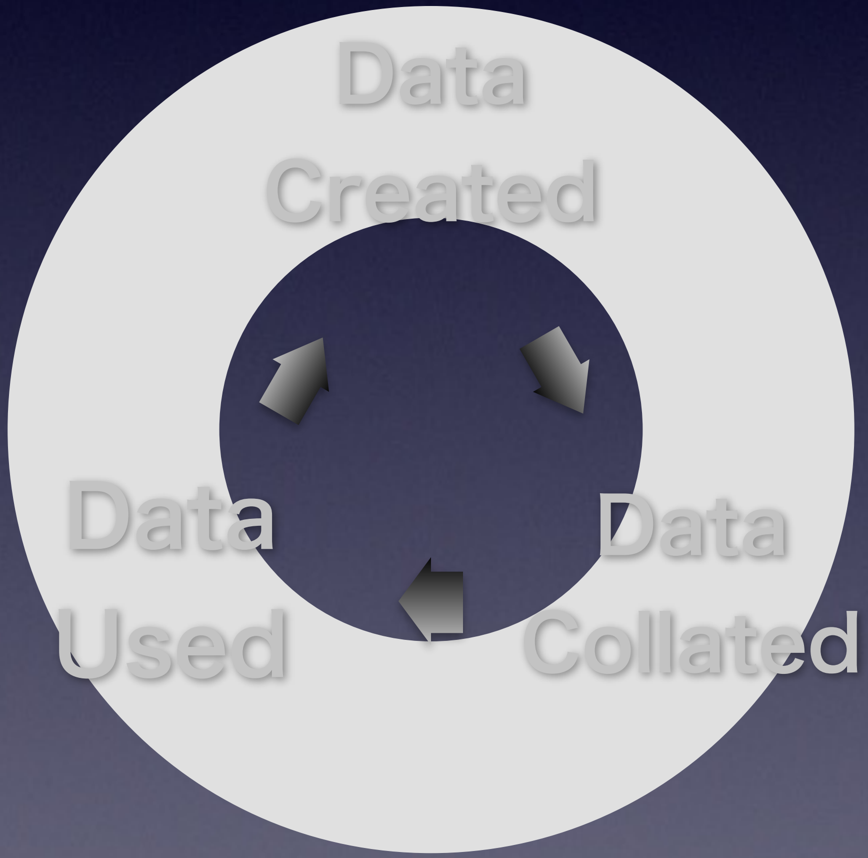
Short Program	User Focus	2-Steps
Fast Development	Efficient Communication	Accurate Spec Design

From the "Unicage Development Method Technical Overview", 2013 USP Lab.

Data Flow

Input Script

POS
Order Data
Master Record, etc.



Output Script

Screen
Report, etc.

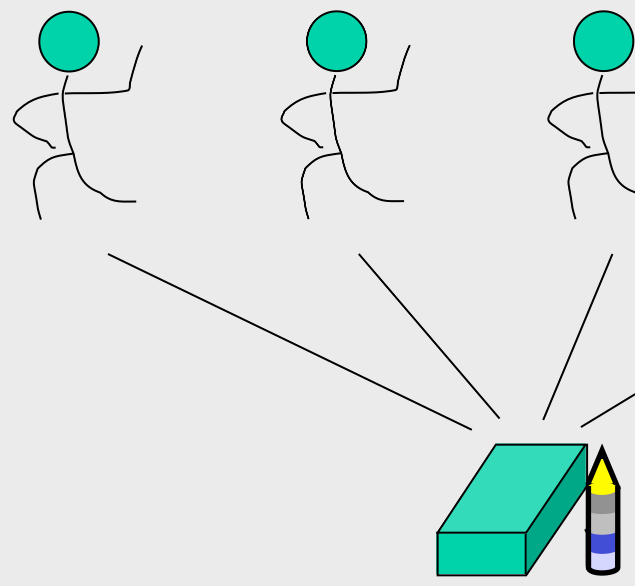
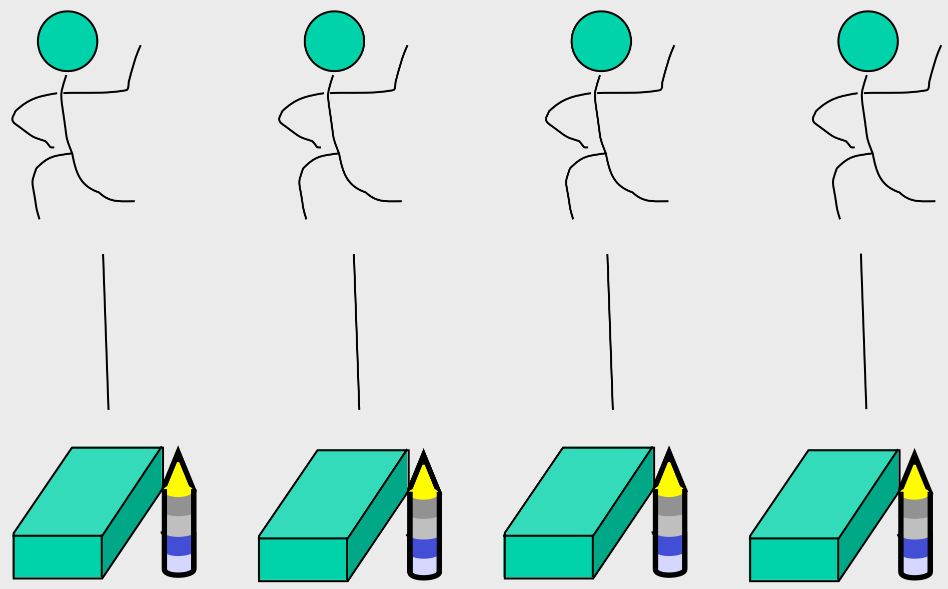
Update/Collate Script

Data merged
5W1H Collation
5 Layer Data Management, etc.

All three systems are created with shell scripts
Data transfer is all performed with File I/F

From the "Unicage Development Method Technical Overview", 2013 USP Lab.

Unicage Development Method
Data Strategy: Distributed

Traditional "Sharing" (centralized)	Full Sharing (distributed)
	
One change of spec affects everyone	One change of spec doesn't affect others
High Load / Program is Complex	Low Load / Simple Program

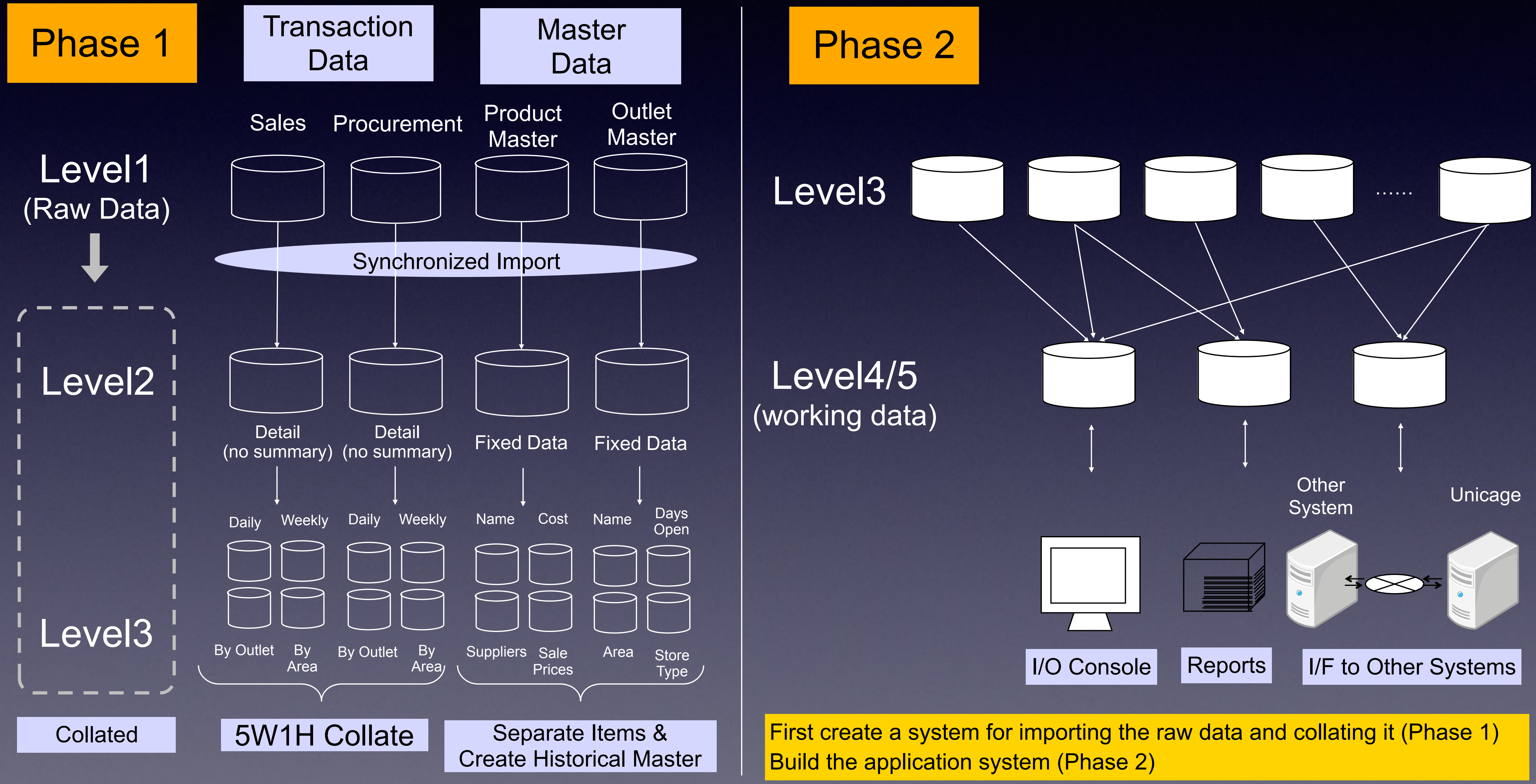
From the "Unicage Development Method Technical Overview", 2013 USP Lab.

Unicage Development Method
5 Layers Data Management

Level 1 (Event Data)	Raw Data (Unsynchronized) Data created as the information is produced (Must not be lost; replicated and backed-up robustly)
Level 2 (Confirmed Data)	Detail Data (Synchronized) Raw data that has been formally imported into the system at a particular point in time
Level 3 (5W1H Data)	Level 2 data is organized and summarized in 5W1H units (Often called “Ultra-organization.” Does not depend on the application.
Level 4 (Application Data)	Data formatted to be easily processed by the application
Level 5 (Output Data)	Report Image Data (Excel, PDF) or Log

From the “Unicage Development Method Technical Overview”, 2013 USP Lab.

Unicage Development Method
5 Layers Data Management



First create a system for importing the raw data and collating it (Phase 1)
Build the application system (Phase 2)

From the "Unicage Development Method Technical Overview", 2013 USP Lab.

Universal Shell Programming Laboratory

五
矢 隹
止
usp lab.

Universal Shell Programming Laboratory, Ltd.

- April 2005 established, Japan
- http://www.usp-lab.com/
- President is Nobuaki TOUNAKA / 當仲寛哲
- fastest-growing enterprise systems development small-middle size software vendor
- sales accounting system, payroll accounting system, corporate system, CRM system, merchandising system, enterprise system self-manufacture etc

Customers

株式会社
良品計画
IDÉE

Takayanagi
Quality Life Partner

おいしんぼ・カラオケ
Pasela

マチのほっとステーション
LAWSON

WORLD

三省堂書店
BOOKS SANSEIDO

ANA FESTA

TRIAL

SUPERMARKET
成城石井

THE BODY SHOP.

LOTTERIA

CO-OP
コープネット事業連合

It's BONANZA CITY
ヨシヅヤ

日本酒類販売株式会社

CREATIVE
Space
TANAKA
LIFE SHOP

CREATIVE LIFE STORE
TOKYU HANDS

花王

THE JAPAN AGRICULTURAL NEWS
日本農業新聞

Hankyu
HANSHOKU

カメラの
キタムラ

USP Lab
main customers

- ウエルシアホールディングス株式会社、全日空商事株式会社、株式会社良品計画、株式会社ワールド、株式会社ローソン、株式会社阪食、株式会社成城石井、株式会社義津屋、株式会社東急ハンズ、株式会社ロッテリア、株式会社キタムラ、株式会社ニュートン、株式会社日本農業新聞、株式会社トライアルカンパニー、日本酒類販売株式会社、株式会社タカヤナギ、株式会社三省堂書店、田中商事株式会社

  etc

  Welcia Holdings, ANA FESTA, Ryohin Keikaku,World, Lowson, Hanshoku, Seijoishii,Yoshiduya,Tokyu Hands, Lotteria, Kitamura, Newton, Nihon Nougyo Shinbun,Trial company, Nihon Shuruihanbai, Takanagi, Sanseido,Tanakashouji etc

USP Lab
shell programming research

- NEDO (New Energy and Industrial Technology Development Organization) - Practical fast information treatment by unicage development method and pipeline calculator
- Tokyo University Tamai lab - Enterprise information system self-manufacture
- Keio University Ohiwa lab - Unicage development method and Japanese programming
- Waseda University Yamana lab - Shellscripting fast data treatment method on multicore processor
- Nagoya University Kawaguchi lab - Emergency data treatment system development by Unicage development method

Drill-down into Unicage

```
% head -5 *
==> FOOD_MASTER <==
000001 Vegetable
000002 Fish
000003 Meet
000004 Beverage

==> SALES <==
000001 20140516130102 000001 235 3
000001 20140516130103 000002 923 4
000001 20140516130103 000003 524 1
000001 20140516130105 000001 625 1
000001 20140516130106 000002 223 3

==> STORE_MASTER <==
000001 Tokyo
000002 Ottawa
000003 Sunnyvale
000004 Cambridge
000005 Sofia
%
```

```
% cat SALES | awk '$2>=20140516130000&&
            $2<20140516140000' | head
000001 20140516130102 000001 235 3
000001 20140516130103 000002 923 4
000001 20140516130103 000003 524 1
000001 20140516130105 000001 625 1
000001 20140516130106 000002 223 3
000001 20140516130106 000003 335 1
000001 20140516130108 000001 754 3
000001 20140516130109 000001 112 8
000001 20140516130112 000002 321 1
000001 20140516130112 000002 441 1
%
```

```
% cat SALES | awk '$2>=20140516130000&&
          $2<20140516140000' | delf 2 |
          head
000001 000001 235 3
000001 000002 923 4
000001 000003 524 1
000001 000001 625 1
000001 000002 223 3
000001 000003 335 1
000001 000001 754 3
000001 000001 112 8
000001 000002 321 1
000001 000002 441 1
%
```

```
% cat SALES | awk '$2>=20140516130000&&
          $2<20140516140000' | delf 2 | awk
          '{print $1,$2,$3*$4}' | head
000001 000001 705
000001 000002 3692
000001 000003 524
000001 000001 625
000001 000002 669
000001 000003 335
000001 000001 2262
000001 000001 896
000001 000002 321
000001 000002 441
%
```

```
% cat SALES | awk '$2>=20140516130000&&
          $2<20140516140000' | delf 2 | awk
          '{print $1,$2,$3*$4}' | sort -
          k1,2 | sm2 1 2 3 3
000001 000001 10146
000001 000002 10312
000001 000003 5191
000001 000004 2529
000002 000001 10146
000002 000002 10312
000002 000003 5191
000002 000004 2529
000003 000001 10146
000003 000002 10312
000003 000003 5191
000003 000004 2529
%
```

```
% cat SALES | awk '$2>=20140516130000&&
          $2<20140516140000' | delf 2 | awk
          '{print $1,$2,$3*$4}' | sort -
          k1,2 | sm2 1 2 3 3 | join1 key=1
          STORE_MASTER -
000001 Tokyo 000001 10146
000001 Tokyo 000002 10312
000001 Tokyo 000003 5191
000001 Tokyo 000004 2529
000002 Ottawa 000001 10146
000002 Ottawa 000002 10312
000002 Ottawa 000003 5191
000002 Ottawa 000004 2529
000003 Sunnyvale 000001 10146
000003 Sunnyvale 000002 10312
000003 Sunnyvale 000003 5191
000003 Sunnyvale 000004 2529
%
```

```
% cat SALES | awk '$2>=20140516130000&&
          $2<20140516140000' | delf 2 | awk
          '{print $1,$2,$3*$4}' | sort -
          k1,2 | sm2 1 2 3 3 | join1 key=1
          STORE_MASTER - | self 3 2 4
000001 Tokyo 10146
000002 Tokyo 10312
000003 Tokyo 5191
000004 Tokyo 2529
000001 Ottawa 10146
000002 Ottawa 10312
000003 Ottawa 5191
000004 Ottawa 2529
000001 Sunnyvale 10146
000002 Sunnyvale 10312
000003 Sunnyvale 5191
000004 Sunnyvale 2529
%
```

```
% cat SALES | awk '$2>=20140516130000&&
          $2<20140516140000' | delf 2 | awk
          '{print $1,$2,$3*$4}' | sort -
          k1,2 | sm2 1 2 3 3 | join1 key=1
          STORE_MASTER - | self 3 2 4 |
          cjoin1 key=1 FOOD_MASTER -
000001 Vegetable Tokyo 10146
000002 Fish Tokyo 10312
000003 Meet Tokyo 5191
000004 Beverage Tokyo 2529
000001 Vegetable Ottawa 10146
000002 Fish Ottawa 10312
000003 Meet Ottawa 5191
000004 Beverage Ottawa 2529
000001 Vegetable Sunnyvale 10146
000002 Fish Sunnyvale 10312
000003 Meet Sunnyvale 5191
000004 Beverage Sunnyvale 2529
%
```

```
% cat SALES | awk '$2>=20140516130000&&
          $2<20140516140000' | delf 2 | awk
          '{print $1,$2,$3*$4}' | sort -
          k1,2 | sm2 1 2 3 3 | join1 key=1
          STORE_MASTER - | self 3 2 4 |
          cjoin1 key=1 FOOD_MASTER - | self
          3 2 4
Tokyo Vegetable 10146
Tokyo Fish 10312
Tokyo Meet 5191
Tokyo Beverage 2529
Ottawa Vegetable 10146
Ottawa Fish 10312
Ottawa Meet 5191
Ottawa Beverage 2529
Sunnyvale Vegetable 10146
Sunnyvale Fish 10312
Sunnyvale Meet 5191
Sunnyvale Beverage 2529
%
```

```
% cat SALES | awk '$2>=20140516130000&&
            $2<20140516140000' | delf 2 | awk
            '{print $1,$2,$3*$4}' | sort -k1,2 |
            sm2 1 2 3 3 | join1 key=1 STORE_MASTER
            - | self 3 2 4 | cjoin1 key=1
            FOOD_MASTER - | self 3 2 4 | sm4 1 1 2
            2 3 3
Tokyo Vegetable 10146
Tokyo Fish 10312
Tokyo Meet 5191
Tokyo Beverage 2529
Tokyo @@@@@@@@@ 28178
Ottawa Vegetable 10146
Ottawa Fish 10312
Ottawa Meet 5191
Ottawa Beverage 2529
Ottawa @@@@@@@@@ 28178
Sunnyvale Vegetable 10146
Sunnyvale Fish 10312
Sunnyvale Meet 5191
Sunnyvale Beverage 2529
Sunnyvale @@@@@@@@@ 28178
%
```

```
% cat SALES | awk '$2>=20140516130000&&
            $2<20140516140000' | delf 2 | awk
            '{print $1,$2,$3*$4}' | sort -k1,2 |
            sm2 1 2 3 3 | join1 key=1 STORE_MASTER
            - | self 3 2 4 | cjoin1 key=1
            FOOD_MASTER - | self 3 2 4 | sm4 1 1 2
            2 3 3 | sm5 1 2 3 3
Tokyo Vegetable 10146
Tokyo Fish 10312
Tokyo Meet 5191
Tokyo Beverage 2529
Tokyo @@@@@@@@@ 28178
Ottawa Vegetable 10146
Ottawa Fish 10312
Ottawa Meet 5191
Ottawa Beverage 2529
Ottawa @@@@@@@@@ 28178
Sunnyvale Vegetable 10146
Sunnyvale Fish 10312
Sunnyvale Meet 5191
Sunnyvale Beverage 2529
Sunnyvale @@@@@@@@@ 28178
@@@@@ @@@@@@@@@ 84534
%
```

```
% cat SALES | awk '$2>=20140516130000&&
             $2<20140516140000' | delf 2 | awk '{print
             $1,$2,$3*$4}' | sort -k1,2 | sm2 1 2 3 3 |
             join1 key=1 STORE_MASTER - | self 3 2 4 |
             cjoin1 key=1 FOOD_MASTER - | self 3 2 4 |
             sm4 1 1 2 2 3 3 | sm5 1 2 3 3 | comma 3 |
             awk '{print $1,$2,"$"$3}' | keta
    Tokyo Vegetable $10,146
    Tokyo      Fish $10,312
    Tokyo      Meet  $5,191
    Tokyo  Beverage  $2,529
    Tokyo @@@@@@@@@ $28,178
   Ottawa Vegetable $10,146
   Ottawa      Fish $10,312
   Ottawa      Meet  $5,191
   Ottawa  Beverage  $2,529
   Ottawa @@@@@@@@@ $28,178
Sunnyvale Vegetable $10,146
Sunnyvale      Fish $10,312
Sunnyvale      Meet  $5,191
Sunnyvale  Beverage  $2,529
Sunnyvale @@@@@@@@@ $28,178
    @@@@@ @@@@@@@@@ $84,534
%
```

```
#!/bin/sh
tmp=/tmp/$$
lv3=$(dirname $0)/../LV3; lv5=$(dirname $0)/../LV5

# header
echo "|===Store===| |===Foods===| |===Sales===|"> $tmp-header
# total, subtotals sales
cat $lv3/SALES                                  | # obtain sales tran
awk '$2>=20140516130000&&$2<20140516140000'     | # select target slot
delf 2                                          | # remove unneeded field
awk '{print $1,$2,$3*$4}'                       | # subtotal for each items
sort -k1,2                                      | # sort for next treatment
sm2 1 2 3 3                                     | # subtotal
join1 key=1 $lv3/STORE_MASTER -                 | # join store names
self 3 2 4                                      | # reorder and select for next
cjoin1 key=1 $lv3/FOOD_MASTER -                 | # join food names
self 3 2 4                                      | # reorder and select
sm4 1 1 2 2 3 3                                 | # subtotal for each stores
sm5 1 2 3 3                                     | # total
comma 3                                         | # comma insertion
awk '{print $1,$2,"$"$3}'                       | # dollar mark insertion
cat                                             > $tmp-sales
# sales report
cat $tmp-header $tmp-sales                      |
keta                                            > $lv5/SALES_REPORT_20140516_13.TXT

rm -r $tmp-*
```

```
%./SHL/MAKE_SALES_REPORT.SH
% cat ./LV5/SALES_REPORT_20140516_13.TXT
|===Store===| |===Foods===| |===Sales===|
        Tokyo     Vegetable       $10,146
        Tokyo          Fish       $10,312
        Tokyo          Meet        $5,191
        Tokyo      Beverage        $2,529
        Tokyo     @@@@@@@@@       $28,178
       Ottawa     Vegetable       $10,146
       Ottawa          Fish       $10,312
       Ottawa          Meet        $5,191
       Ottawa      Beverage        $2,529
       Ottawa     @@@@@@@@@       $28,178
    Sunnyvale     Vegetable       $10,146
    Sunnyvale          Fish       $10,312
    Sunnyvale          Meet        $5,191
    Sunnyvale      Beverage        $2,529
    Sunnyvale     @@@@@@@@@       $28,178
        @@@@@     @@@@@@@@@       $84,534
%
```

some tips

- CPITAL FILE/DIRETORY_NAMES for parmanent files/directories
- lower case files/directory_names for temporary files/directories
- Use files instead of variables
- Use commands instead of functions

A4 paper x 1

- Good Unicage shellscript (application) within size of an A4 paper
- At longest, it should be within 2 A4 papers
- If your shellscript is over thousands, it is completely wrong
  It may be a time to discuss to create new commands to correct your wrong

GET Unicage

- OSS rewritten version "Open usp Tukubai"
- However, Open usp Tukubai is a little bit buggy
- Assess: Amazon EC2 Tukubai environment
- Assess: Download Free Trial Version

  Open en.usp-lab.com and click "GET UNICAGE NOW!"

column: ttt development method

- From the beginning, ONGS's approach is similar to Unicage
- ttt development method is base on our original method, adding Unicage LV1~LV5 concept, advanced commands and *BSD based directories structure.

Conclusion

- buzzword xRAD is getting raised these days
- Unicage is Shellscripts and Commands based enterprise system development method
- Unicage is logical and already proven in real enterprise system development field
- This approach is great for *BSD