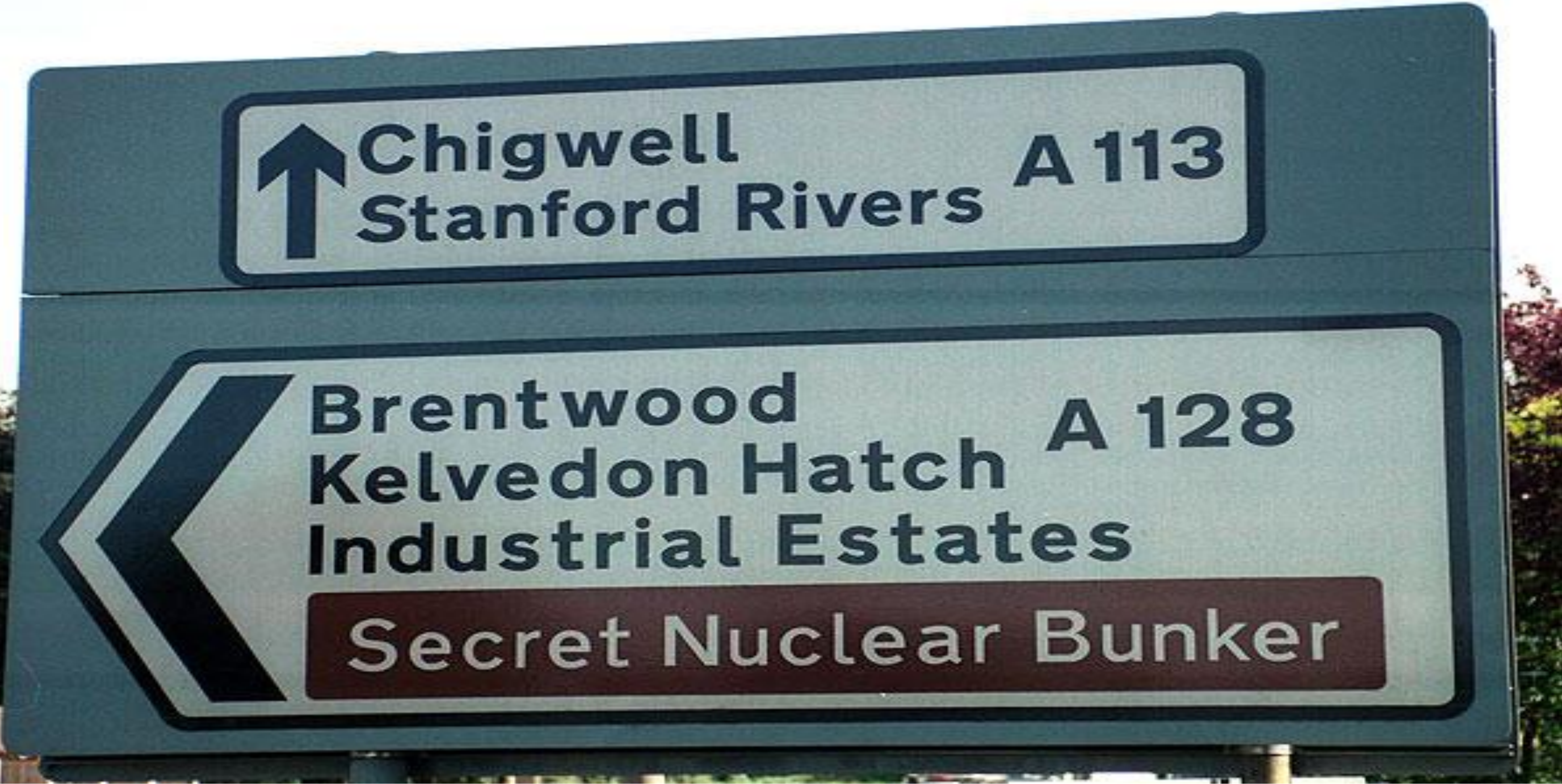
PUG CHALLENGE EXCHANGE AMERICAS

Tales of the Secret Bunker 2016 (231) Dump and Load Edition

Mike Furgal – Director MDBA and Pro2 Services
Gus Bjorklund - Lackey

find directions where location-name = "secret bunker"



Type II storage areas have lessened the need to do regular dump and load processing, however you still need to dump and load to maintain good performance. The frequency of the dump and load tends to be every 3 to 5 years.

The Bunkerteers took it upon themselves to test various dump and load technologies

What needs to be dumped?

- Data Definitions
- Data
- Sequence Current Values
- Security
  - _user
  - SQL Permissions
- Audit Rules
- Proutil Describe
- Etc

Before you dump

- Backup the database
- Verify the backup
  - Best approach would be to restore the database
- Run a tabanlys
  - Used to compare after the load to make sure we got all the data
- Make a note of:
  - db block size, ai block size, bi block size, code page, etc
  - who has DBA rights
- DO YOU HAVE ENOUGH DISK SPACE ?

How to Load

- Data Definitions
- Data
- Sequence Current Values
- Security
  - _user
  - SQL Permissions
- Audit Rules
- May be more

After You Load

- Backup the database
- Run a tabanlys
  - Compare the row count to the original
- Proutil Describe
  - Compare it to the original
  - Bi Blocksize, BI Cluster Size, AI Blocksize, etc
- Enable After Imaging
- Rebaseline OE Replication

what could go wrong?

mike

Data Dumping and Loading Options

ASCII Options

- Ascii Dump through the dictionary
- Ascii Load through the dictionay
  - With and without active indexes
- Bulkload

Buffer Copy

- Buffer Copy from one database to another

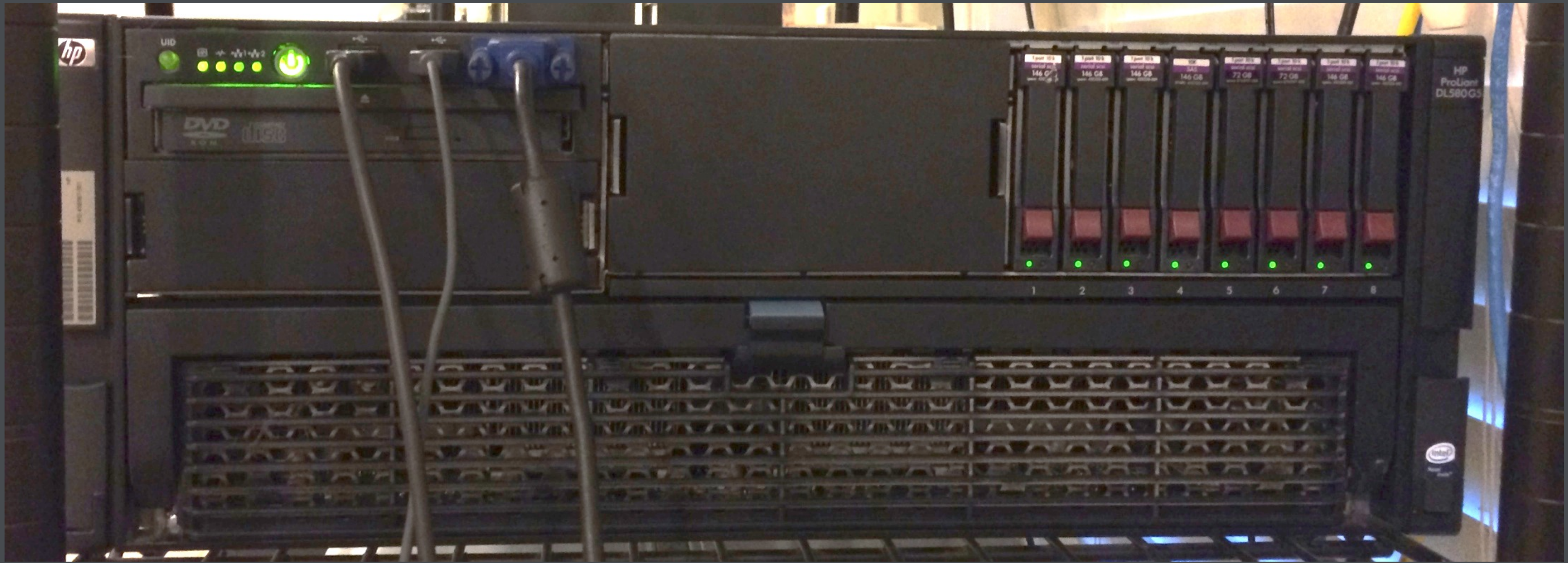
Binary Options

- Binary Dump
- Binary Load

On to the Tests

EDEN.MILL
ST ANDREWS
BUNKER
BREW
DOUBLE
I.P.A
ALC.
7.2%
VOL
330
ML ℮

bunker



Bunker Machine

- 4 quad-core 2.4 GHz Intel processors
  - 4800.25 bogomips
- 64 GB memory
- 8 x 146 GB 10,000 rpm sas drives
  - 2 RAID 10
  - 6 RAID 0 for /opt/tmp
- 16 x 300 GB 10,000 rpm drives
  - RAID 10 for /opt/db
- 8 x 300 GB 10,000 rpm drives
  - RAID 10 for /opt/db1
- Centos Linux 6.7
- OpenEdge 11.5.1

New this machine costs $35,000 USD.

Used we found it for $3,500 USD

BIGROW test runs in 4 seconds
24 MB/Second

Database Statistics

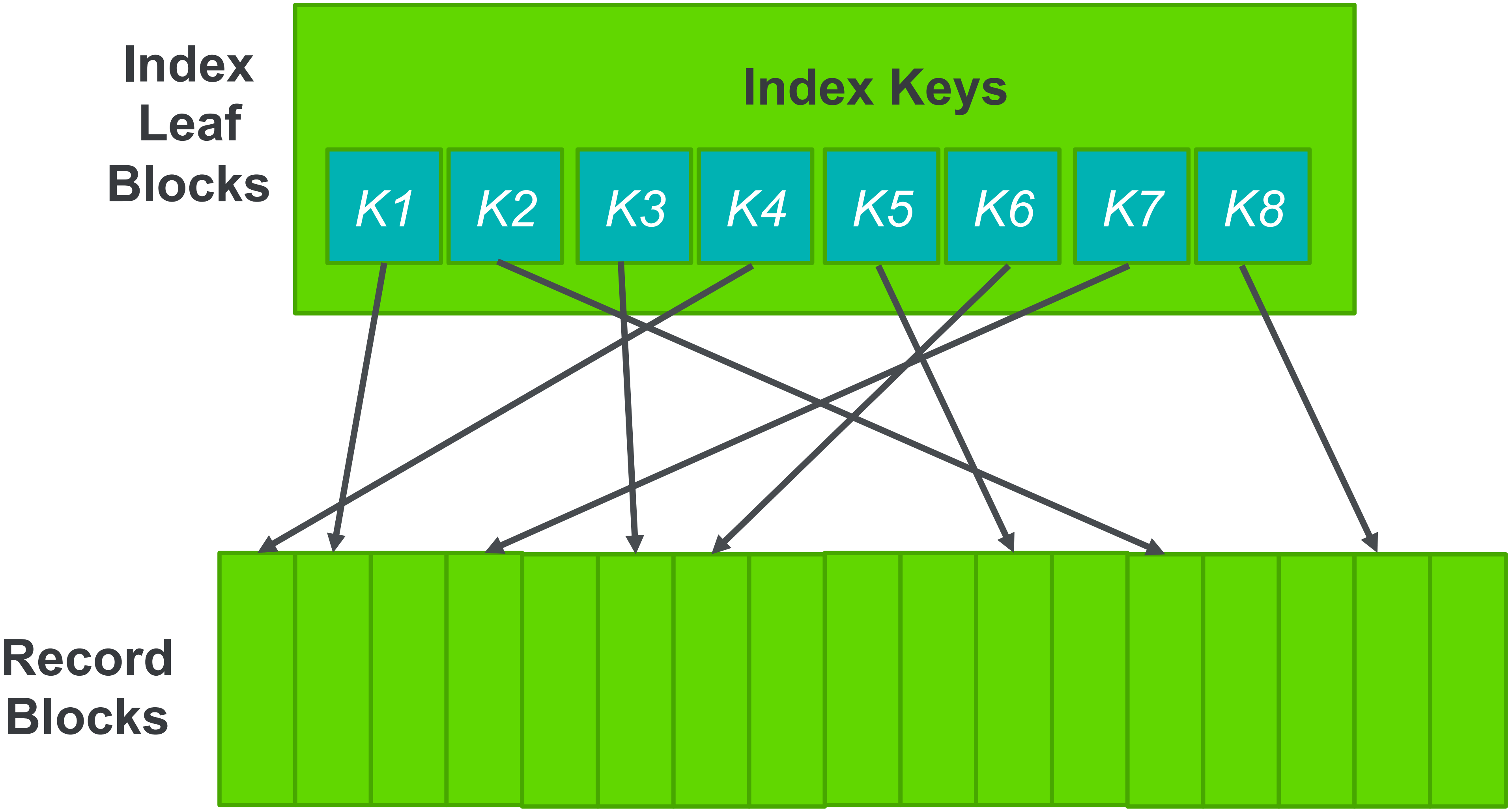
- Size: 36 GB
- Tables: 835
- Indexes: 1,736
- Areas
  - 49 Data Areas
  - 49 Index Areas

Table	Rows	Size
Table 1	13,495,717	4.6G
Table 2	52,307,552	3.5G
Table 3	1,873,601	2.1G
Table 4	2,432,884	1.3G
Table 5	9,430,367	1007.8M

gus

ASCII^H^H^H^HText Dump

- Using the data dictionary to dump the data programatically
  - by running prodict/dump_d.p
- Where to put dump files?
  - Same filesystem 86:12
  - Different filesystems 85:33
- Making the filesystem cache smaller made a slight difference
  - 86:12 vs 87:46



key order does not match record storage order

Scatter Matters

- The database used was freshly dumped and loaded. To get past this, we ran a scatter analysis program which looks at the logical scatter of the database
- Changed the primary index on the top 43 largest tables
  - Row counts were greater than 500,000
  - % of rows logically scattered > 10
- Comparison
  - Non scattered dump: 87:46
  - Scattered dump: 133:21 --- 52% slower!! This is important!

Text Load

- Using the dictionary load programmatically with prodict/load_d.p
  - Loading into preallocated space: 376:20
  - Loading into variable extents: 365:06
  - Loading from a different filesystem: 354:15
- Loading no active indexes: 276:04
  - Loading the data: 231:34
  - Index rebuild: 44:33
- Bulkload: 110:05
  - Loading the data: 65:32
  - Index rebuild: 44:33

Best Text Dump and Load Result
233:09
3 hours, 53 minutes, 9 seconds

By:

Dictionary Text dump,
Bulkload, and
Index Rebuild

mike

Buffer Copy

- The approach here is to connect to thre source database and buffer-copy all the data to the target database.
- Tests performed
  - Single User
  - Multi-User
  - With active Indexes
  - No active indexes
  - Parallel

```
FOR EACH s.<table> NO-LOCK:
    BUFFER-COPY s.<table> to t.<table>.
END.
```

Buffer Copy Results

Buffer Copy	
Single User	257:80
Single User no index	134:31 + 44:33 = 179:01
Single User no index scattered	179:36 + 44:33 = 224:06
Multi User Scattered no index	193:11 + 44:33 = 237:44
Multi User Scattered by area parallel	130:21
Multi User no index by area parallel	**82:39 + 44:33 = 127:09**

The parallel processing used 8 processors running at the same time

How to Parallelize the process

- A bunch of sophisticated analysis was done to optimize this process.

BRUTE

How to Parallelize the process

- A bunch of sophisticated analysis was done to optimize this process.
- Take the size of the largest table and use that as a guide
- Combing all the table into their respecive areas
- Break them up into N sized units
  - Where N is the size of the largest table

- This process is very complex and borders on rocket science

The longest thread took 130 minutes
The shortest thread took 68 minutes

gus

Binary Dump

- Build scripts to dump out all the data.
- Tests
  - Single Users
  - Read Only
  - Multi user
  - Parallel

Binary Dump	
Dump non-scattered data	10:17
Dump Scattered	46:20
Dump non-scattered -index 0	8:34
Dump Scattered -index 0	8:35

It’s 4x slower to dump the scattered data

Using -index 0 does a table scan. While this is the fasted dump method, the order of the rows will not be useful for most applications

Notice the similar times for the non-scattered and scattered table scan dumps. Even the (expert?) bunkerteers do stupid experiments

Binary Dump Results

Binary Dump	
Single user	46:20

Binary Dump Results

Binary Dump	
Single user	46:20
Single User with large -B and lruskips	44:13
Multi User with large -B and lruskips	**42:46**
Single User with -RO	45:07
Single User with -RO with large -B and lruskips	43:10

Much special equipment was needed

Binary Dump Results

Binary Dump	
Single user	46:20
Single User with large -B and lruskips	44:13
Multi User with large -B and lruskips	42:46
Single User with -RO	45:07
Single User with -RO with large -B and lruskips	43:10
Parallel MU by Area with large -B and lruskips	**27:45**
Parallel -RO by area with large -B and lruskips	28:44
Parallel MU By Area -Bp 64 with large -B and lruskips	29.14

mike

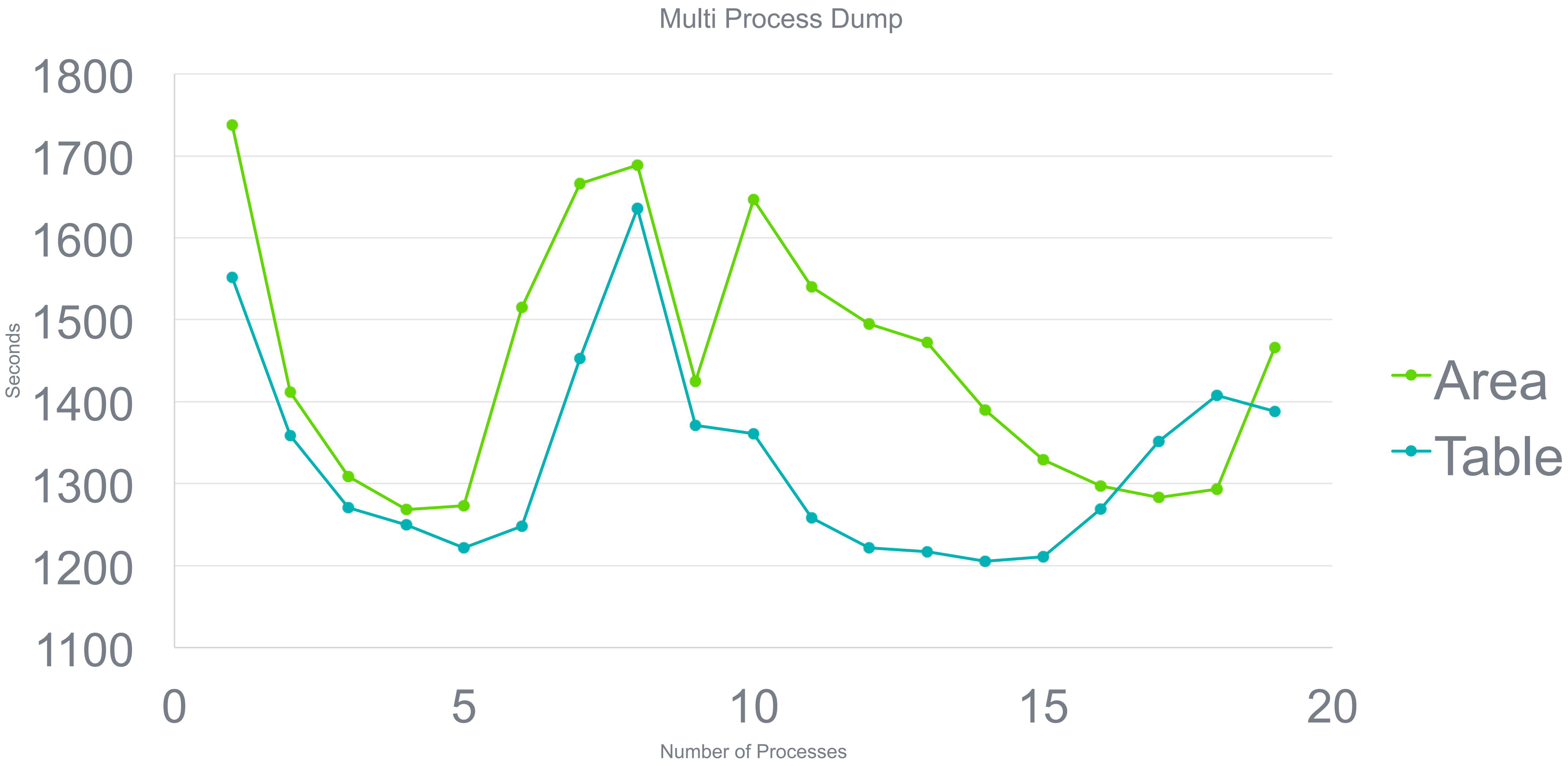
A Simple Parallel Processor

```
# Thread Scheduler

for i in `cat tables`
do
    currentThread=`ls -1 *.working 2> /dev/null | wc –l`
    if [ $currentThread -le $threads ]
    then
        ./dump_table.sh $i > $i.out &
    else
        while [ 1 ]
        do
            currentThread=`ls -1 *.working 2> /dev/null | wc –l`
            if [ $currentThread -le $threads ]
            then
                break
            fi
            sleep 0.1
        done
        ./dump_table.sh $i > $i.out &
    fi
    sleep 0.2
done
wait
```

```
# Worker Thread

echo $$ > $$.working
proutil scattered -C dump $1 ./bdump
rm -f $$.working
```


Multi Process Dump
1800
1700
1600
1500
1400
1300
1200
1100
Seconds
0
5
10
15
20
Number of Processes
Area
Table

gus

Binary Load

- Binary load tests include
  - Single User
  - Multi User
  - Parallel loads by area
  - Single user with build indexes
  - Multi user with build indexes
  - Parallel load by area with build indexes

Binary Load Results

Binary Load	
Single User	36:17 + 44:33 = 80:50
Multi User	20:52 + 44:33 = 65:28
Parallel Multi User	17:27 + 44:33 = 62:00
Single User Build indexes	75:41
Multi User build indexes	61:40
Parallel Multi User build indexes	**46:49**

The best dump and load result is:

Parallel dump by table	20:22
Parallel load by area and build indexes	46:49
Total time excluding backups, etc	67:06

Results Summary

Results		
Slowest Round Trip	•Dictionary Dump •Dictionary Load •Indexes Active	133:21 367:20 500:41 (**8:20:41)**
Fastest TEXT	•Dictionary Dump •Bulkload •Index Rebuild	133:21 65:32 44:33 243:26 (**4:03:26)**
Fastest Buffer Copy	•Parallel by area •Indexes Inactive	82:39 44:33 127:09 (**2:07:09)**
Fastest Binary	•Parallel dump by table •Parallel Load build indexes	20:22 46:49 67:06 (**1:07:06)**

BEER BUNKER

PUGCHALLENGE EXCHANGE
AMERICAS