#SHAREorg

SHARE
Technology · Connections · Results

Dramatically Reduce the Cost of Sequential File Accesses in CICS

Stephen Reid
Antares Computing Pty Ltd

August 10, 2012
Session Number 12029

SHARE in Anaheim 2012

Agenda

- Background
- Requirements
- Solution
- Implementation
- Refinements and Extensions
- Making the Solution Universally Applicable
- Questions

Background

- It all started with 9/11
- FBI mandate to screen all financial transactions
- 15 million SWIFT transactions per day
- Typically ~50 fields of ~100 characters, per transaction
- Need to check each field against every suspect name
- Fuzzy match on 20,000 names initially – and growing!
- Benchmark showed impossible with normal access methods
- Asked to design/develop a **super efficient** data access
- >500% faster than required access speed
- Fuzzy match algorithm a story in itself – for another time . . .

Requirements

- Read the “Next Record” with minimum machine instructions
- Allow multiple (unlimited) simultaneous Read accesses
- Avoid “Below-the-Line” storage overheads
- Avoid Open/Close overheads (x15 million)/day
- (Allow flexibility in Record Length)

Possible Extra Requirements (not for FBI)

- The following functions could introduce Threadsafe issues:
  (colour-coded blue in subsequent slides)
- Support real-time Updates, Additions, and Deletions (ESDS)
- Ensure any changes are controlled and secure
- Ensure data is always Current
- Prevent “Double Updates”

Solution

- Main Memory ! (20,000 X 100 bytes = only 2M)
- Allocate a Linked List of Record “Cells” Above the 16M Line
- Store Control Information in a CICS Table (32 byte CSECT)
- Make Control Table “Resident”, so never freed
- Resident means it occupies only 32 bytes, not 4K
- Preload the file during PLTPI
- Access Method only involved **once** at CICS Startup
- Subsequent “READ” of each Record just moves its address
- If CICS dies, PLTPI simply reloads the file on restart
- Changes performed through a single common routine

Implementation

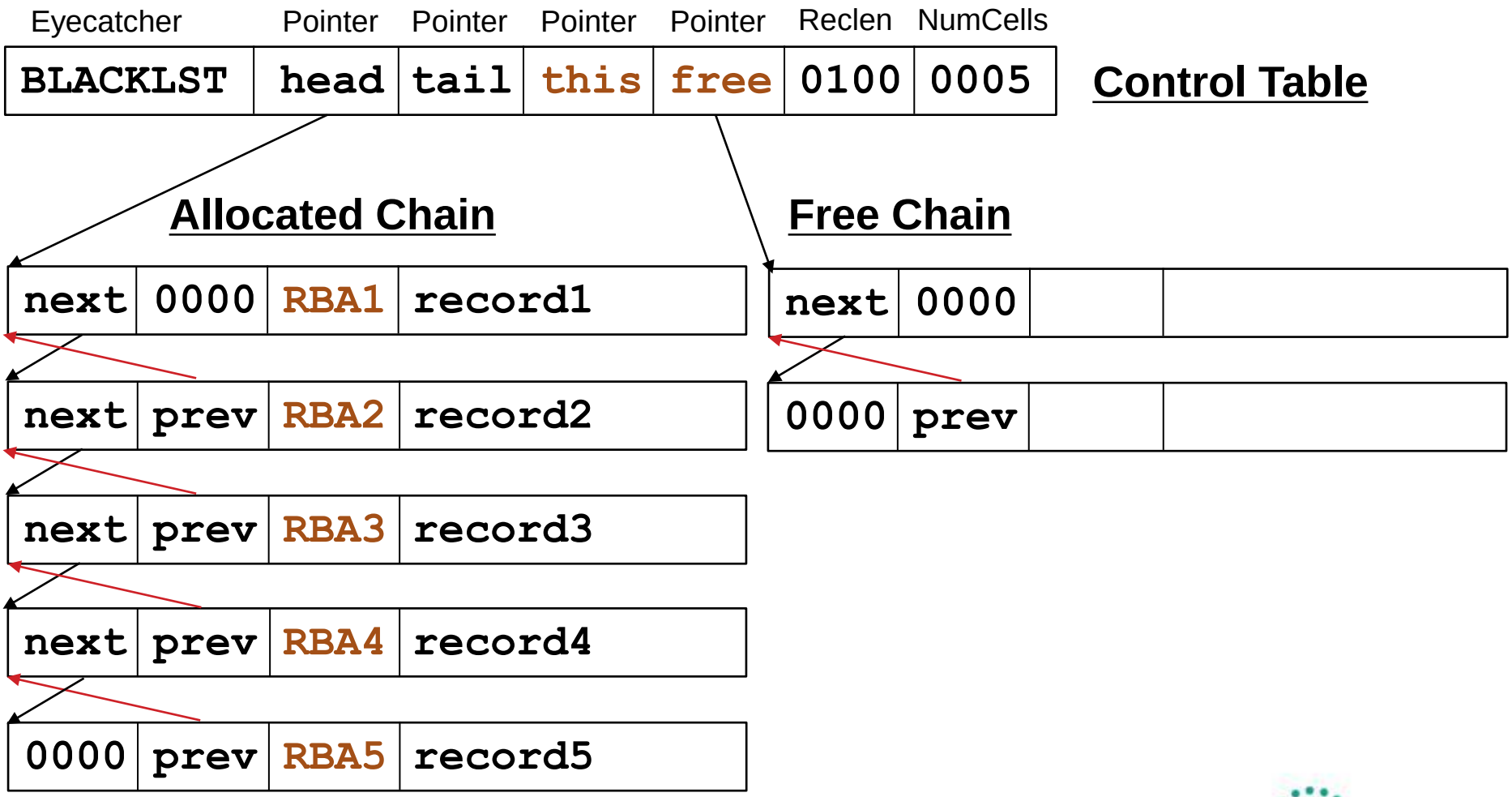
- Define a PLTPI program to LOAD the Control Table and READ all the records into the Linked List
- Each program that wants to READ the “file” just LOADs the Control Table and runs the Linked List
- All Updates, Additions and Deletions CALL a common subroutine to perform the function (for ESDS, not QSAM)
- Updates ENQ on the RBA, and update in place
- Additions write to the end of the file, and add the new cell to the end of the Linked List
- Deletions free the cell for subsequent Additions, and use CONTROL access on the ESDS to physically update the CI

Implementation

The following Control Table is defined for each Linked List:

```
         TITLE 'CONTROL TABLE FOR LINKED LIST OF SWIFT MESSAGE FIELDS.'
BLACKLST CSECT
*****************************************************************
* DEFINITION OF THE CONTROL TABLE FOR THE LINKED LIST OF 'BLACK NAMES'.
* IT SHOULD BE DEFINED TO CICS AS RES=YES SO IT IS NEVER FREED,
* IS LOADED ONLY AT CICS STARTUP, AND OCCUPIES ONLY 32 BYTES.
*****************************************************************
BLACKLST RMODE ANY
BLACKLST AMODE 31
TABLNAME DC    CL8'BLACKLST'    TABLE NAME EYECATCHER FOR DUMP
HEADPTR  DC    XL4'FF000000'    ADDRESS OF FIRST CELL IN ALLOCATED CHAIN
TAILPTR  DC    XL4'FF000000'    ADDRESS OF LAST CELL IN ALLOCATED CHAIN
THISPTR  DC    XL4'FF000000'    ADDRESS OF CURRENT CELL IN THE CHAIN
FREEPTR  DC    XL4'FF000000'    ADDRESS OF FIRST AVAILABLE FREE CELL
CELLLEN  DS    F'100'           LENGTH OF EACH CELL'S DATA AREA
CELLNUM  DS    F'0'             NUMBER OF CURRENTLY ALLOCATED CELLS
         END
```

Implementation



Implementation

Then it is defined in the application program as follows:

```
LINKAGE SECTION.
01   Filename-CTRL.                                  <-(For example)
         05      List-Name       PIC X(8).          <-(useful in a dump)
         05      Head-PTR        POINTER.
         05      Tail-PTR        POINTER.
         05      This-PTR        POINTER.
         05      Free-PTR        POINTER.
         05      Cell-Len        PIC S9(8) COMP.
         05      Cell-Num        PIC S9(8) COMP.
```

Implementation

And for each Linked List, the Cell is defined as:

```
01      This-Cell.
            05      Next-PTR        POINTER.
            05      Prev-PTR        POINTER.
            05      This-RBA        PIC S9(8) COMP.      <- for ESDS only
            05      This-Data.
                        10    Whatever is needed.
```

Implementation

So the program simply performs the following:

```
EXEC CICS LOAD
          PROGRAM (Filename)
          SET (ADDRESS OF Filename-CTRL)
END-EXEC
```

Do not move any values to any of the fields in Filename-CTRL
These will all be pre-initialized by the PLTPI program.

Implementation

Then “Read” and process each record as follows:

```
SET ADDRESS OF This-Cell TO Head-PTR
PERFORM UNTIL ADDRESS OF This-Cell IS NULL
        Process This-Data
        ,
        ,
        SET ADDRESS OF This-Cell TO Next-PTR
END-PERFORM
```

We can also process the List in reverse (LIFO) order by using Tail-PTR and Prev-PTR instead of Head-PTR and Next-PTR

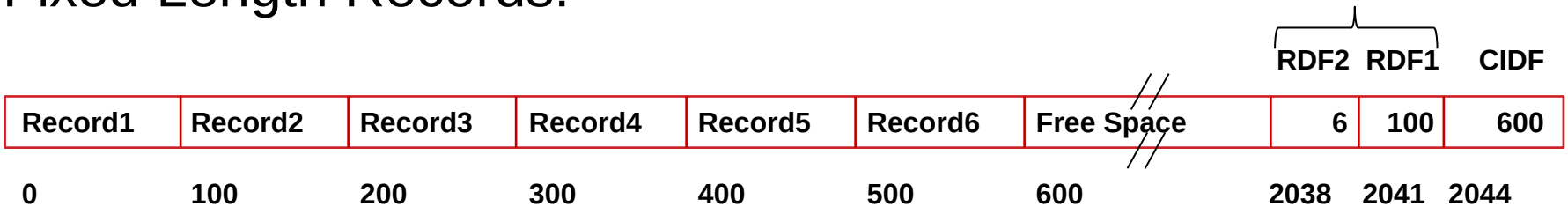
Refinements and Extensions

- If an ESDS is to be updated then define the dataset profile with CONTROL access so CI can be manipulated directly

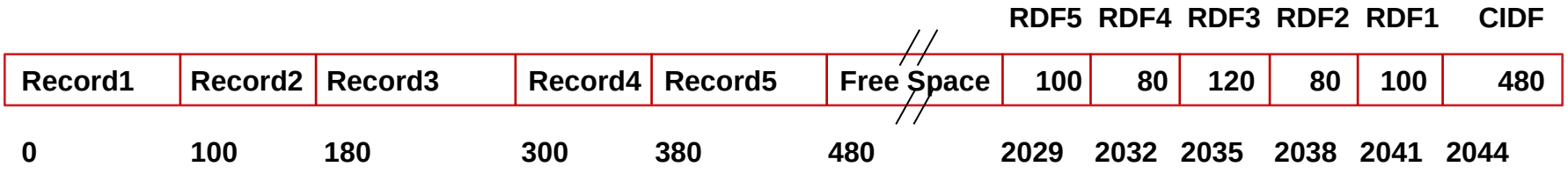
Refinements and Extensions

ESDS Control Interval

Fixed Length Records:



Variable Length Records:



Refinements and Extensions

- If an ESDS is to be updated then define the dataset profile with CONTROL access so CI can be manipulated directly
- Since ESDSs are not officially recoverable, any changes must be logged if forward or backward recovery is required
- Since all records are available to all tasks, to ensure consistency, we should move our record to working-storage if we execute any CICS commands during our use of it **
- If we DON'T execute any CICS commands within the loop described on the previous slide, then an occasional SUSPEND command would avoid a possible runaway task
- Use 64 bit addressing and put the DATA above the bar, just keep the linked list of ADDRESSES below the bar

Refinements and Extensions

For 64 bit, the Control Table defines a Linked List of Addresses,
and the records are moved down below the bar as required.

```
01      This-Cell.
            05      Next-PTR        POINTER.
            05      Prev-PTR        POINTER.
            05      This-RBA        PIC S9(8) COMP.     <- for ESDS only
            05      This-Len        PIC S9(8) COMP.     <- length of data
            05      This-Addr       PIC X(8).           <- 64 bit Address
            05      Curr-PTR        POINTER.            <- 0 if not below_
            05      Curr-CTR        PIC S9(4) COMP.     <- current # in use
```

with the data defined as:

```
01      This-Data.
            05      Whatever is needed.
```

Refinements and Extensions

And we “Read” and process each record as follows:

Note: This is NOT Threadsafe!

```
SET ADDRESS OF This-Cell TO Head-PTR
PERFORM UNTIL ADDRESS OF This-Cell IS NULL
        IF Curr-PTR IS NULL
        THEN CALL MoveDown USING ADDRESS OF This-Cell
        ELSE ADD 1 TO Curr-CTR  END-IF
        SET ADDRESS OF This-Data TO Curr-PTR
        Process This-Data
        CALL FreeUp USING ADDRESS OF This-Cell
        SET ADDRESS OF This-Cell TO Next-PTR
END-PERFORM
```

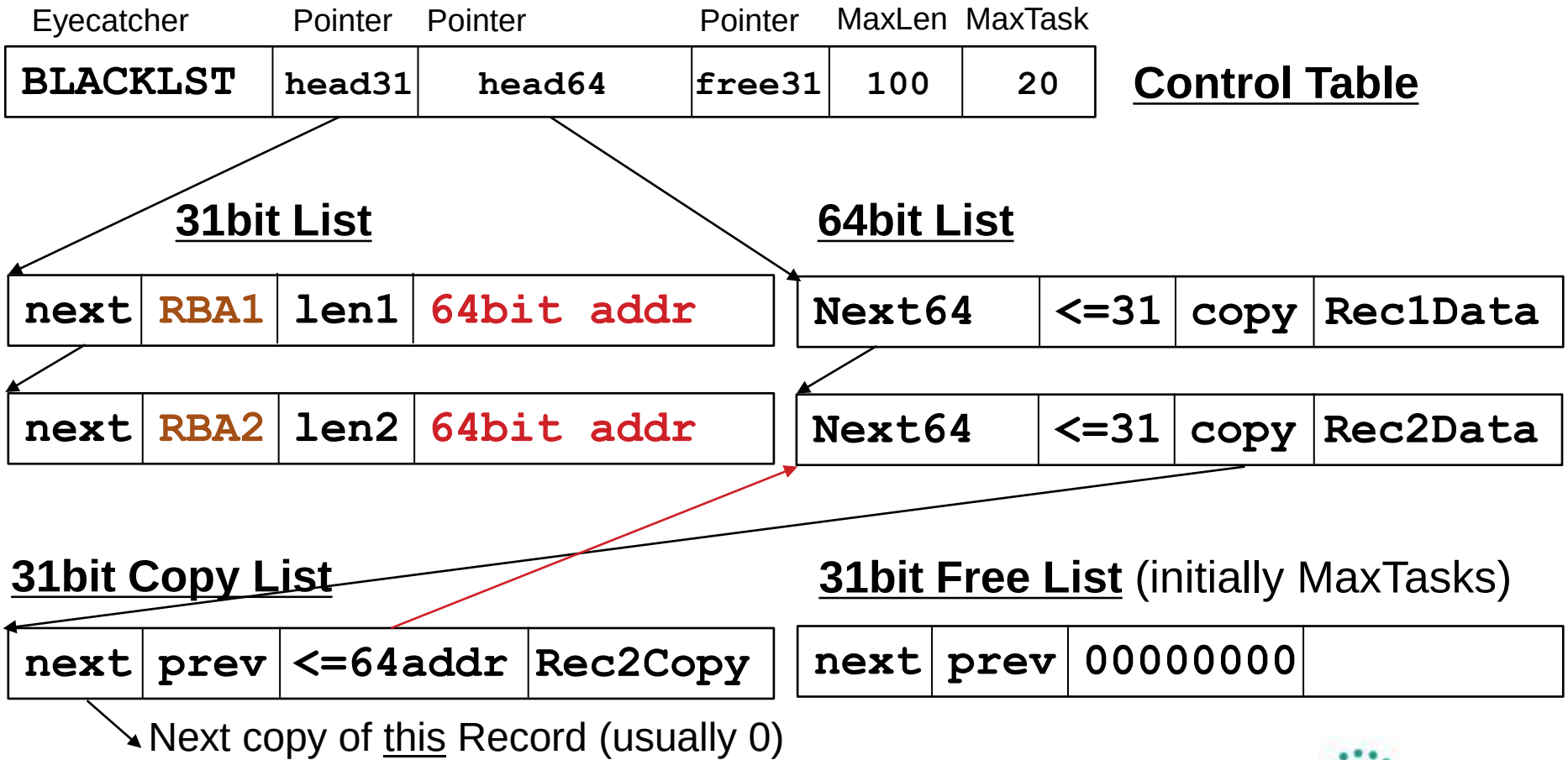
Refinements and Extensions

To make this Threadsafe we need to avoid any possibility of more than one task accessing anything at the same time:

```
SET ADDRESS OF This-Cell TO Head-PTR
PERFORM UNTIL ADDRESS OF This-Cell IS NULL
        CALL MoveDown USING ADDRESS OF This-Cell,
                            ADDRESS OF This-Data
        Process This-Data
        CALL FreeUp USING ADDRESS OF This-Data
        SET ADDRESS OF This-Cell TO Next-PTR
END-PERFORM
```

Refinements and Extensions

And then comes the interesting bit !



Making the Solution Universally Applicable

- Define ESDS in CICS FCT (CSD) with CONTROL access
- Assemble & Link the Corresponding CSECT into DFHRPL
- Define that “Program” in CICS PPT (CSD) as RESIDENT
- Define the Linked-List Loading Program in PLTPI
- Filename is passed as a Parameter to Loading Program
- Everything is defined by the File-specific Control Table
- Functional Routines are all generic
- Use Compare and Swap (CSG) to make 64-bit threadsafe
- If you would like any help with any of these techniques, call me on +61-414-SPREID or +1-925-452-6456, or email me at StephenPReid@gmail.com

Questions?