Improving the performance of web-initiated CICS transaction workloads

Patrick Fournier
Serge Bourlot
John Bachiochi
(SysperTec)

Tuesday, August 13, 2013 (12:15 to 1:15 PM)
Session # 13456
Presenters’ Bio

• Patrick Fournier
  • Modernization of IBM mainframe applications
  • Last 5 years: web-enablement of 3270 applications
  • Initiator of CICS performance improvement R&D project

• Serge Bourlot
  • z/OS system product developer (custom TS)
  • Strong z/OS internals - especially multi-tasking services
  • Lead developer for R&D project

• John Bachiochi
  • Develop and support VSE and MVS SW products
  • Part of benchmarking team for R&D project
Changing Nature of CICS TP Workloads

A growing number of consumers conduct their personal business and shop in the Cloud every day, which results in enormous flows of web-initiated transactions hitting the supporting – oftentimes CICS – applications.
How It All Started …

- Two web-enablement prospects with similar exponentially-growing web-initiated CICS transaction workloads:
  - Improve web access (middleware) to CICS apps
  - Tens of million of CICS transactions per day
  - Web-initiated traffic reaching CICS apps as web services
  - Clients + consumers in “self-service” mode via web portals
  - Small number of query transactions \( \rightarrow 85\% \) TP workload
  - No direct revenue generation: part of expected “service”
  - Cost of doing business today: improved user experience
Changing Nature of CICS TP Workloads

CICS Transactions

3270 Screens

Web Services

TCP/IP

Web Services

Web Portals
C/S Apps
(WUI/GUI)

3270 Terminal Emulation

Trained Employees + Agents

Clients + Consumers (Self-Service)

Accessibility
Convenience
Revenues
Labor costs

CPU Usage
Mainframe costs
Capacity
Response times
“We run several thousand transactions per second now that our systems are opened up to the Internet and mobile devices.”

“The number of look-to-book transactions for our hotel chain has changed from 8 – 10 searches for one booking to potentially thousands of searches for one booking.”

“We expect more growth coming from the mobile channel and we also foresee a workload increase from new self-service applications.”

“In our enterprise architecture, the mainframe is our transaction processing box; it is optimized for transaction processing, and we expect it to be further optimized for this in the future.”

We have moved most of our services to stateless web services, which expose core transactions directly.”
## CICS: Original Design vs. Current Usage

<table>
<thead>
<tr>
<th></th>
<th><strong>ORIGINAL DESIGN</strong></th>
<th><strong>CURRENT USAGE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>When</strong></td>
<td>Early 70s</td>
<td>Today + Tomorrow</td>
</tr>
<tr>
<td><strong>File Organization</strong></td>
<td>ISAM $\rightarrow$ VSAM</td>
<td>VSAM $\rightarrow$ DBMS/SQL (DB2)</td>
</tr>
<tr>
<td><strong>Need for Data Services</strong></td>
<td>Yes</td>
<td>No (DBMS-Provided)</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>BTAM $\rightarrow$ TN3270</td>
<td>HTTP/S over TCP/IP</td>
</tr>
<tr>
<td><strong>Client Device</strong></td>
<td>3270 Workstations</td>
<td>Web-Enabled Devices</td>
</tr>
<tr>
<td><strong>User Interface</strong></td>
<td>Green Screens</td>
<td>WUI/GUI $\rightarrow$ Web Services</td>
</tr>
<tr>
<td><strong>Application Training</strong></td>
<td>Yes</td>
<td>No (Self-Service)</td>
</tr>
<tr>
<td><strong>Main User Population</strong></td>
<td>Staff + Agents/BP</td>
<td>Clients + Consumers</td>
</tr>
<tr>
<td><strong>Total Users</strong></td>
<td>Thousands</td>
<td>Millions</td>
</tr>
<tr>
<td><strong>Concurrent Users</strong></td>
<td>Hundreds</td>
<td>Thousands</td>
</tr>
<tr>
<td><strong>Transaction Volumes</strong></td>
<td>10s x Thousands</td>
<td>Millions</td>
</tr>
</tbody>
</table>
CICS Overhead Considerations

CICS adds significant overhead to web-initiated traffic which it was not designed for. How much overhead and why?
How Much System Overhead?

- Systems (CICS, LE …) = most CPU consumption
- Transaction programs = very little CPU consumption
Where Does CICS Overhead Comes From?

- Event-level TCP/IP listener → costly CICS disruptions
- Presentation layer gets activated by web service calls
- VSAM-intended data services get activated for DBMS
- Not designed for z/OS PE multi-tasking services
Multi-Tasking 101

- MVS multi-tasking:
  - Event Control Blocks (ECB)
  - Pause Elements (PE):
    - Introduced with z/OS
    - Integrated to z/OS lower system layers (control block redesign)
    - IBM → Use PE over ECB for efficiency and performance
    - PE = tool of choice for multitasking subsystems

- CICS multi-tasking:
  - Double-level multitasking: CICS → MVS
  - Relies upon ECB (not IBM-recommended PE)
Transaction Server Prototype

What would a new transaction server specifically designed to run web-initiated CICS transactions with best possible response times, throughput and footprint look like?
Transaction Server Prototype - Design Brief

- Design brief for low-overhead TS prototype:
  - No screen UI: web service only UI
  - Web interface through HTTP/S server with:
    - Message-level (rather than event-level) TCP/IP listener
    - Web service management facility: scripts and scenarios
    - zIIP for parser (XML, SOAP …) and other CPU-intensive tasks
    - RACF server-level (rather than user-level) authentication path
  - Efficient PE-based multi-tasking
  - New API: run existing CICS programs under new TS
  - Direct DBMS/SQL calls + reuse existing DBMS data services
  - VSAM data services
We already had all the components we needed between VIRTEL and VIRSV (except VSAM data services): all we needed was to couple them!
Benchmarking

What kind of response times, throughput and footprint is this new transaction server capable of?
Two versions of **same** program(s):
- One running under CICS/TS
- One running under V/TS

**Compare:**
- Response-time
- CPU consumption
- Throughput
Scenario #1 – Query Only

- Queries = heaviest (costliest) TP workload = best ROI
- Program used for benchmark:
  - COBOL + DB2
- Business Logic
  - Retrieve user contact data from database
- RACF activation:
  - CICS/TS: first incoming call only (cached authentication)
  - V/TS: each incoming call (no cached authentication)
  - Need to configure Virtel STC to cache RACF control blocks
### Scenario #1 - CPU Consumption

<table>
<thead>
<tr>
<th>QUERY-ONLY</th>
<th>CICS/TS</th>
<th>V/TS</th>
<th>CPU Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/IP</td>
<td>48.71</td>
<td>7.37</td>
<td>84.87%</td>
</tr>
<tr>
<td>TP Monitor</td>
<td>427.08</td>
<td>115.32</td>
<td>72.99%</td>
</tr>
<tr>
<td>DB9xxxxxxx</td>
<td>25.18</td>
<td>.69</td>
<td>97.26%</td>
</tr>
<tr>
<td>RRS</td>
<td>.25</td>
<td>.06</td>
<td>76%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>501.22</strong></td>
<td><strong>123.44</strong></td>
<td><strong>75.37%</strong></td>
</tr>
</tbody>
</table>
## Scenario #1 - Overall Performance

<table>
<thead>
<tr>
<th>QUERY-ONLY</th>
<th>CICS/TS</th>
<th>V/TS</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG Response Time</td>
<td>327</td>
<td>46</td>
<td>7 times shorter</td>
</tr>
<tr>
<td>AVG Throughput</td>
<td>30</td>
<td>152</td>
<td>5.4 times larger</td>
</tr>
<tr>
<td>MAX Throughput</td>
<td>35</td>
<td>223</td>
<td>6.4 times larger</td>
</tr>
<tr>
<td>CPU Consumption</td>
<td>501</td>
<td>123</td>
<td>4 times smaller</td>
</tr>
</tbody>
</table>
Scenario #2 – All I/O Types

- Updates = 10% or less of TP workload = worse ROI
- Program used for benchmark:
  - COBOL + DB2
- Business Logic
  - Create new user
  - Retrieve new user from database
  - Update new user’s telephone number
  - Update new user’s qualifier
  - Retrieve updated user from database
  - Delete updated user from database
- Work in progress (Not optimized yet)
### Scenario #2 - Overall Performance

<table>
<thead>
<tr>
<th>MIXED I/O TYPES</th>
<th>CICS/TS</th>
<th>V/TS</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG Response Time</td>
<td>284</td>
<td>142</td>
<td>2 times shorter</td>
</tr>
<tr>
<td>AVG Throughput</td>
<td>35</td>
<td>66</td>
<td>1.9 times larger</td>
</tr>
<tr>
<td>MSU Consumption</td>
<td>21465</td>
<td>10194</td>
<td>2.1 times lower</td>
</tr>
</tbody>
</table>
Comparing Scenarios #1 vs. #2 Results

<table>
<thead>
<tr>
<th></th>
<th>QUERY-ONLY</th>
<th>MIXED I/Os</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG Response Time</td>
<td>7 x shorter</td>
<td>2 x shorter</td>
</tr>
<tr>
<td>AVG Throughput</td>
<td>5.4 x larger</td>
<td>1.9 x larger</td>
</tr>
<tr>
<td>MSU Consumption</td>
<td>4 x lower</td>
<td>2.1 x lower</td>
</tr>
</tbody>
</table>

- **Updates take longer** than queries → TS execution time reduction applies to smaller % of overall response time → overall response time reduction is smaller
- CPU-intensive **DB2 update locking** runs inside TS address space = TS CPU reduction applies to lower % of overall CPU consumption → overall CPU reduction is smaller
- Updates typically only about 10% of all I/O but scenario #2 includes as many updates as queries → worse case scenario
- Scenario #2 not optimized yet (contentions …) → Results might improve
- Current scenario #2 results still = excellent improvement
Program Changes

What kind of changes must be applied to CICS programs to run under V/TS? Could those changes be automated?
Program Changes – Overview

- Interface with transaction server (V/TS vs. CICS):
  - Parameter list
  - Processing flow
  - Initialization section
  - Core section
  - Termination section

- File and DBMS accesses:
  - Direct DB2/SQL calls
  - [VSAM calls]
V/TS Parameter List:

- Address of input data area
- Length of input data area
- Address of output data area
- Length of output data area
- Return code

COPY VSVCLIST.

<table>
<thead>
<tr>
<th></th>
<th>Feature</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>LIST-POINTER.</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>LIST-ADR-REQU</td>
<td>USAGE IS POINTER.</td>
</tr>
<tr>
<td>03</td>
<td>LIST-ADR-REQUL</td>
<td>USAGE IS POINTER.</td>
</tr>
<tr>
<td>03</td>
<td>LIST-ADR-RESP</td>
<td>USAGE IS POINTER.</td>
</tr>
<tr>
<td>03</td>
<td>LIST-ADR-RESPL</td>
<td>USAGE IS POINTER.</td>
</tr>
<tr>
<td>03</td>
<td>LIST-ADR-RC</td>
<td>USAGE IS POINTER.</td>
</tr>
</tbody>
</table>
Program Changes – Processing Flow

CICS/TS

PROCEDURE DIVISION.

START-OF-PROGRAM.

1. PERFORM INITIATE-PROCESS THRU INITIATE-PROCESS-XIT.

2. PERFORM MAIN-PROCESS THRU MAIN-PROCESS-XIT.

3. PERFORM TERMINATE-PROCESS THRU TERMINATE-PROCESS-XIT.

EXEC CICS RETURN
END-EXEC.

Repeat transaction processing:
✓ Was handled by CICS
✓ Now handled by program

V/TS

PROCEDURE DIVISION.

PERFORM UNTIL NO-MORE-PROCESS

1. PERFORM INITIATE-PROCESS THRU INITIATE-PROCESS-XIT

2. PERFORM MAIN-PROCESS THRU MAIN-PROCESS-XIT

3. PERFORM TERMINATE-PROCESS THRU TERMINATE-PROCESS-XIT

END-PERFORM.

Z000-EXIT.
GOBACK.

1. Initialization
2. Core Processing
3. Termination
Program Changes - Initialization Section

**CICS/TS**

INITIATE-PROCESS.

EXEC CICS ADDRESS EIB(DFHEIBLK)
END-EXEC.

MOVE SPACES TO WORK-DFHCOMMAREA.
MOVE EIBCALEN TO EIBCALEN-DECIMAL.
IF EIBCALEN-DECIMAL > MAX-COMMAREA
   MOVE RC-TOO-LONG-COMMAREA TO WORK-RETURN-CODE
   MOVE MSG-TOO-LONG-COMMAREA TO
   WORK-ERROR-MESSAGE
   GO TO INITIATE-PROCESS-XIT
ELSE
   MOVE DFHCOMMAREA TO WORK-DFHCOMMAREA.
   MOVE 0 TO WORK-RETURN-CODE.
   MOVE SPACES TO WORK-ERROR-MESSAGE.

INITIATE-PROCESS-XIT.
EXIT.

**V/TS**

2000-CALL-VSVPSYNC SECTION.

CALL 'VSVPSYNC' USING REFERENCE POINTER-OF-POINTER
RETURNING CODE-PSYNC-NUM
IF CODE-PSYNC-NUM NOT ZERO
   GO TO 2099-FIN
END-IF.

2030-SET-VIRSV-POINTERS.

SET ADDRESS OF LIST-POINTER TO POINTER-OF-POINTER
SET ADDRESS OF REQU TO LIST-ADR-REQU
SET ADDRESS OF REQUL TO LIST-ADR-REQUL
SET ADDRESS OF RC TO LIST-ADR-RC.

2030-SET-SCENARIO-POINTERS.

SET ADDRESS OF VTSADD-ACTION TO REQU-ACTION-PTR
MOVE VTSADD-ACTION TO WORK-ACTION
SET ADDRESS OF VTSADD-COMMAND TO REQU-COMMAND-PTR
MOVE VTSADD-COMMAND TO WORK-COMMAND
SET ADDRESS OF VTSADD-DATA-WORK-AREA TO REQU-DATA-WORKAREA-PTR
SET ADDRESS OF VTSADD-RETURN-CODE TO REQU-RETCODE-PTR
SET ADDRESS OF VTSADD-ERROR-MESSAGE TO REQU-ERROR-MSG-PTR.

2099-FIN.
EXIT.

Copy input data:

- From COMMAREA
- To local working storage area
Program Changes – Core Section

CICS/TS

B110-RECEIVE-MAP.

PERFORM B111-DO-THE-DB2-WORK
THRU B111-DO-THE-DB2-WORK-EXIT.
MOVE WORK-DFHCOMMAREA TO VTSCOMM.
MOVE DATA-WORK-AREA TO PRESET-DWA.
EXEC CICS WRITEQ TS
QUEUE (PRESET-OPTIONS-QNAME)
FROM (PRESET-OPTIONS-QUEUE)
LENGTH (PRESET-OPTIONS-LENGTH)
ITEM (PRESET-OPTIONS-ITEM)
MAIN
END-EXEC.

B110-RECEIVE-MAP-EXIT.
EXIT.

V/TS

B110-RECEIVE-MAP.

PERFORM B111-DO-THE-DB2-WORK
THRU B111-DO-THE-DB2-WORK-EXIT.
MOVE WORK-DFHCOMMAREA TO VTSCOMM.
IF VIRSV-PROCESS
MOVE DATA-WORK-AREA TO VTSDWA
GO TO B110-RECEIVE-MAP-EXIT.
MOVE DATA-WORK-AREA TO PRESET-DWA.

B110-RECEIVE-MAP-EXIT.
EXIT.

Simple EXEC CICS replacement:
✓ Replace Temporary Storage Queue logic with equivalent V/TS logic
✓ LINK to V/TS through dynamic calls
Program Changes – Termination Section

CICS/TS

TERMINATE-PROCESS.

MOVE WORK-COMMAREA TO DFHCOMMAREA.

TERMINATE-PROCESS-XIT.
EXIT.

V/TS

TERMINATE-PROCESS.

MOVE DATA-WORK-AREA TO VTSDWA.
MOVE WORK-RETURN-CODE TO VTSRC.
MOVE WORK-ERROR-MESSAGE TO VTSMSG.

TERMINATE-PROCESS-XIT.
EXIT.

Copy output data:

✔ From local working storage area
✔ To COMMAREA
**Program Changes – DBMS Accesses**

**CICS/TS**

EXECUTE-DB2-FILE.

EXEC SQL PREPARE DYNAMSELECT
   FROM :COMMAND-STATEMENT
   END-EXEC.
PERFORM PROCESS-ERROR-CODE
   THRU PROCESS-ERROR-CODE-XIT.
IF HBWORK-RC-SUCCESS
   NEXT SENTENCE
ELSE
   GO TO EXECUTE-DB2-FILE-XIT.
EXEC SQL EXECUTE DYNAMSELECT
   USING :HOLD-WORK-AREA-ACTUAL
   END-EXEC.

EXECUTE-DB2-FILE-XIT.
EXIT.

**V/TS**

EXECUTE-DB2-FILE.

EXEC SQL PREPARE DYNAMSELECT
   FROM :COMMAND-STATEMENT
   END-EXEC.
PERFORM PROCESS-ERROR-CODE
   THRU PROCESS-ERROR-CODE-XIT.
IF HBWORK-RC-SUCCESS
   NEXT SENTENCE
ELSE
   GO TO EXECUTE-DB2-FILE-XIT.
EXEC SQL EXECUTE DYNAMSELECT
   USING :HOLD-WORK-AREA-ACTUAL
   END-EXEC.

EXECUTE-DB2-FILE-XIT.
EXIT.

**Direct access to DB2/SQL:**
- ✓ No change to application code
- ✓ Use IBM standard DB2/SQL protocol and data services
- ✓ Avoid redundant CICS data services
Program Changes – Automation Potential

• Initial onetime code modification:
  • Manual at start → develop some experience
  • Partly automated later on?

• Techniques to avoid dual maintenance:
  • Automated conversion tools → pre-compilation
  • Hide code differences behind copybooks/includes
  • Insert dual logic for execution-time

• Most maintenance changes will fall outside TS interface
Implementation Strategy

Suggested Implementation
Key Implementation Considerations
Suggested Implementation

CICS/TS Transactions

Equivalent V/TS Transactions

3-6 high-volume transactions generate 80% of overall footprint
Key Implementation Considerations

- Limited scope
  - 3-6 transactions generate 80% of web-initiated transaction volume
  - Don’t replace or eliminate CICS: create alternate path
- Risk management
  - Retain CICS transaction path for fallback and (debug) reference
- Avoid dual maintenance
  - Automate code changes with pre-compiler
  - Hide changes that cannot be automated behind copy/includes
  - Most maintenance changes will fall outside TS interface
- Real world = need VSAM support
  - Currently: DBMS/SQL only (DB2, etc)
  - Short term: use EXCI or VXCI to submit VSAM I/Os through CICS
  - Long term: develop “minimum” VSAM support?
VSAM Support – “Hybrid” Solution

Direct SQL calls to DB2 and other DBMS. Route VSAM calls to CICS via EXCI or VXCI.

- RACF ACF2 TSS
- Database (DB2, etc.)
- Modified CICS Application Program
- VIRSV Multi-Tasking (PE)
- zIIP XML/SOAP Parsing
- Web Service Call
  - TCP/IP Listener
  - HTTP/S Server
  - Authentication
- Scenarios & Scripts
- Data Formatting Templates
- CICS/TS VSAM Access
  - EXCI Multi-command session mgmt.
  - VXCI Mono-command No session mgmt.
- Direct SQL Calls
- TCB
- SRB
Cost/Benefit Analysis

Mainframe organizations are in cost containment mode!

**APPLICATIONS / SYSTEMS:**
- Unfamiliar technology & skillset
- Untested and untrusted
- Code changes
- Duplicate maintenance
- Debugging

**CFO / CIO / OPS MGR:**
- Smaller footprint → reduced MF costs
- Shorter response times → customer satisfaction
- Increased throughput → delayed HW upgrade + growth support
Conclusions and Next Steps

What have we proven and where do we go from here?
Conclusions

• Verified initial overhead assumption:
  • 90-99% of transaction footprint is system (CICS, LE, etc) overhead
• Identified alternative to greatly improve:

<table>
<thead>
<tr>
<th>Query-Only</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response times:</td>
<td>7 times faster</td>
</tr>
<tr>
<td>Throughput:</td>
<td>6 times larger</td>
</tr>
<tr>
<td>Footprint:</td>
<td>4 times smaller</td>
</tr>
</tbody>
</table>

• Expected annual cost savings = $100Ks to $millions
• Main issues:
  • Risk: unfamiliar + untested + untrusted
  • Code modifications + long-term support
  • No VSAM support to date → Limited “real life” applicability
• Cost/benefits analysis
Next Steps

• Confirm DB2/SQL benchmark results with real customer transaction in real production environment
• Add VSAM support
• Then …

The best way to predict the future is to create it

Abraham Lincoln
SHARE expo booth # 521 (VIRTEL)

patrick.fournier@syspertecc.us

+1 (925) 954-9649