Integration of new technology with a workhorse system while preserving scalability and high availability

Colette A. Manoni
Senior Technical Staff Member, IBM
What is a workhorse system?

http://www.merriam-webster.com/dictionary/workhorse definition ...

a dependable machine or vehicle that is used to do a lot of work

My definition ...

The system that sits quietly in the background
May not be the newest or coolest
But it does most of the critical work.

You don’t really notice it ... unless it goes down ...
z/TPF is the workhorse system for airlines and credit cards
The TPF Challenge
Our customers are breaking new world records every day
Can you beat some of these?

As of 3/7/14, it has been
4,567 days since the last
customer visible outage

1.5 million
database reads
per second

3,381,161,400
messages in
24-hours

1-hour sustained
peak I/O of 2,136,571
I/O per second

New in use today

Train conductors view manifest
and lift tickets onboard the train
from mobile devices

Securely encrypt credit
card numbers

© 2014 IBM Corporation
A bit of history ....

stores information from 1,100 reservation desks for every seat in the American Airlines system
TPF: A “purpose-built” solution

- TPF has a long history spanning more than 40 years starting at the birth of “real time” computing representing IBM’s long term commitment to the platform
- Used by some of IBM’s largest customers for mission critical (“revenue bearing”) workloads

IBM History

- 1968
  - PARS
- 1960s
  - Application
- 1924 – CTR becomes International Business Machines
- 1924 – CTR Corp. was incorporated
- 1911 – Computing Tabulating Recording (CTR) Corp. was incorporated

Other industries develop applications for TPF

Most reservation applications derived from this application base

~1/2 of IBM’s History supporting TPF

- 1968
  - IBM
- 1982
  - TPF becomes TPF
- 1989
  - TPF 2.1 (LC support)
- 1994
  - TPF 3.1 (TC support and C Lang.)
- 1994
  - TPF 3.1 (31-bit and virtual addressing)
- 2005
  - z/TPF (64-bit, SOA, security, open run-time/dev.)
Over the years basic principles have remained the same

- **Processing**
  - Short process existence
    - Pre-allocation
    - No paging
  - Non-preemptive scheduling
  - “Soft real-time” system
  - Fast error recovery

- **Data**
  - Large, single-image database, fixed-size records (duped)
  - Horizontal striping - parallel I/O
  - Fine grain locking
  - Caching
  - Memory resident programs

- **Network**
  - Asynchronous
  - Kernel scoped connections
Pre-carved structures

- When system is IPLed all memory is carved up into different structures and put onto lists
  - system work blocks
  - frames for heap areas
  - I/O blocks
  - etc.

- All processes (ECBs) are allocated with fixed addresses
  - shared programs – all reside in memory!
  - private memory (malloc)
  - Shared memory (shm*, system heap, globals)

- When work comes into the system ... grab an item from a list and go

- Lose some flexibility, but gain (a lot!) in performance
Dispatching

- System manages all work through a set of lists ... one for each core or CP
  - Primary lists include
    - Input - new work entering the system
    - Ready - in progress work that is ready to run
    - Defer - lower priority work

- When a message comes into the system it gets put on the Input List
  - As long as there are resources available, messages are processed from the input list
  - Defined threshold triggers input list “shutdown”

- Items get dispatched and are allowed to run until one of following:
  - Give up control by doing an I/O operation - blocked / waiting
  - Give up control voluntarily - defer list
  - Have run for 500 ms - take an error / terminate
Database access

- z/TPF provides for a horizontally striped record database
- Each record can be addressed directly for fast access
- Locking performed at the record level to reduce contention
- I/O requests are distributed across multiple volumes for parallel access
Integration of new technology: TCP/IP

- TCP/IP is primary/base protocol in use today

- Like all other systems, establishing a session between systems incurs a fair amount of overhead
  - example, keep-alive option to reuse session but creating new connections

- Due to the nature of z/TPF - can manage lots of short lived messages
  - messages are what flows over the sockets

- Integrated sockets at the system level
  - Session established at the system level
  - Sockets also managed at the system level
    - Allows an ECB to process a single message and exit

- z/TPF unique extensions
  - Activate on receipt - activate an ECB when a message arrives
  - Activate on accept - activate an ECB when a connection is accepted

- Implement standard API but map to z/TPF’s unique characteristics
Integration of new technology: Apache HTTP server

- Apache HTTP server
  - Long running server model using threads
    - On *nix platform eliminates overhead associated with creating processes
    - But, on z/TPF processes (ECBs) are pre-allocated!
  - Mapped threads to ECBs
  - Each thread (ECB) handles and individual request and returns
    - Request handling matches z/TPF model - short process time
    - But, ECB still associated as part of Apache server (long running)

- Worked well for a number of years, but not optimal use of resources
Integration of new technology: z/TPF HTTP Server

- z/TPF HTTP Server
  - Based on TCP/IP extensions - activate on receipt
  - No need to use threads
  - ECBs activated when message arrives, and exits when request completed
  - ECBs no longer dedicated to server
  - Allows for asynchronous processing when TPF application needs to make calls out to other systems

- Implement standard API but map to z/TPF’s unique characteristics
Integration of new technology: Eventing / Notifications

- Events are one-way communications that some thing has happened
- z/TPF applications have the ability to call a function to signal an event
  - Example: flight time changed send instant message to all passengers
- As we want to keep application processing short, split event processing into 2 parts
  - Collection of the data for the event (application ECB)
  - Format and transmittal of the event (event ECB)
- Once application ECB collects all the raw data for an event, the event is put onto an internal queue
- Another ECB pulls the event from the queue formats the data (XML, etc.) and transmits to consumers

➢ Minimal impact to applications, asynchronous processing model
Integration of new technology: Mobile

- z/TPF is well suited for processing mobile requests:
  - Lots of small messages
  - Asynchronous processing
  - Fast response time

- Integration with mobile devices is through mobile device server such as Worklight and web services
Conclusion

z/TPF has long history of processing very large volumes of transactions with very fast response times, and continues to evolve by integrating and adapting new technologies while maintaining its basic principles that allow it to scale
Thank You