

Centrelink Update

Farewell V7R1, we hardly knew you.

Goff Cureton - May 2010



G'day,

For those of you that don't know me I'm one of Centrelink's M204 DBAs – in particular, I'm their tame M204 internals / assembler guy.

We installed V7R1 + Sirmods 7.5 into production in Feb '09, later upgrading to Sirmods 7.6 in Dec '09. Both combinations proving quite successful and stable for us, enough to be comfortable when recommending them to others. We moved production onto V7R2 + Sirmods 7.7 just recently, at the start of April '10.

Centrelink at a Glance

- Distributes AU\$86.8 billion in social security payments on behalf of policy departments
- Has approximately 6.8 million customers
- Pays about 10.4 million individual entitlements each year
- Is one of the largest employers in the country (~27,000 staff)
- Has more than 1000 service delivery points ranging from large Customer Service Centres to small visiting services
- Offers over 50 online services via the web and automated phone facility



The obligatory overview of Centrelink as an organisation

Current as at April 2010. Figures taken from official public website -->

http://www.centrelink.gov.au/internet/internet.nsf/about_us/facts.htm

Current governmental policy initiatives seeing Centrelink and many other departments being merged and reorganised under a common DHS (Department of Human Services) portfolio. Streamline service delivery, single point of contact, single network, consolidate infrastructure, etc etc.

Centrelink at a Glance (cont)

- Sends over 89 million letters to customers each year
- Conducts more than 4 million reviews each year
- Has more than 650,000 booked office appointments each month
- Grants more than 2.7 million new claims each year
- Receives 33.7 million phone calls each year
- Delivers approximately AU\$365 million a year in payments to customers outside Australia



Centrelink as an organisation overview (part 2). And yes, I'm aware of the irony of have two "at a glance" slides

Current as at April 2010. Figures taken from official public website -->
http://www.centrelink.gov.au/internet/internet.nsf/about_us/facts.htm

Those overseas payments are typically Australian pensions – so don't get your hopes up about an unexpected cheque suddenly arriving in the mail.

A Glance at the Hardware

Canberra :

CDC1	IBM 2097-711 (z10)	PERA, cbre	7,072 mips
CDC2	IBM 2094-735 (z9)	CBRA, CBRC	12,952 mips
CDC3	IBM 2097-714 (z10)	BNEA, CBRB, CBRD, cbrf, xc3	9,319 mips

Bruce :

BDC1	IBM 2094-719 (z9)	SYDA, CBRH	7,951 mips
BDC2	IBM 2094-724 (z9)	MELA, sydb	9,550 mips
BDC3	IBM 2094-720 (z9)	xb3	8,274 mips

Total

55,118 MIPS



Centrelink runs on a distributed sysplex spread across two geographically dispersed sites – one either side of town. Separate power supply (substations, grid etc) and networking etc. Redundant links between the two, taking different routes across town - for the backhoe factor.

Each site has enough MIPS to run our entire production workload. Although we don't run normal hot site / dark site mix. Both are hot, "half loaded" (is that warm?). Seems to be a historical comfort thing more than anything else – if we lose a site we only lose half capacity. But does complicate life with respect to DR strategies. Of course, we have appropriate disaster and capacity on demand agreements with IBM. Thankfully never had to invoke it.

CDC1, CDC2 CDC3, BDC1 etc are simply "names" we attach to the physical machines just to identify them. The numbering reflects their position relative to a particular wall in the data centre. 1 is closest, 3 is furthest away - works just fine. Lpar names in upper case have M204 in some form, lower case is hosting other non-M204 products.

Currently 4 z9's and 2 z10s. A little unbalanced. There's two more z10s already onsite in Bruce, ready for install. Issues with insufficient / inadequate power supply to building. Means we've got to power down two z9s to bring up one z10. Lot's of fun with moving lpar's around. Making another attempt (err, that'd be the 3rd or 4th I think) while I'm here, to upgrade BDC3 to a z10. That's why there's nothing on it at the moment – just the coupling facility. Going to merge it and BDC1 onto a single z10.

A Glance at the Hardware

Canberra :

CDC1	IBM 2097-711 (z10)	PERA, cbre	7,072 mips
CDC2	IBM 2094-735 (z9)	CBRA, CBRC	12,952 mips
CDC3	IBM 2097-714 (z10)	BNEA, CBRB, CBRD, cbrf, xc3	9,319 mips

Bruce :

BDC1	IBM 2094-719 (z9)	SYDA, CBRH	7,951 mips
BDC2	IBM 2094-724 (z9)	MELA, sydb	9,550 mips
BDC3	IBM 2094-720 (z9)	xb3	8,274 mips

Total

55,118 MIPS



CBRA hosts 5 Development and 7 Test onlines.

CBRB is sysprogs O/S install and test., Also my “private” M204 R&D sandbox (good for destructive testing).

CBRC hosts TPNS (nucleus regression testing)

CBRD hosts SAF (applications pre-release regression testing)

CBRE is prod DB2

CBRF is test DB2

CBRH is applications PST (applications load testing)

BNEA is prod onlines B, G and Y

MELA is prod onlines H, J and S

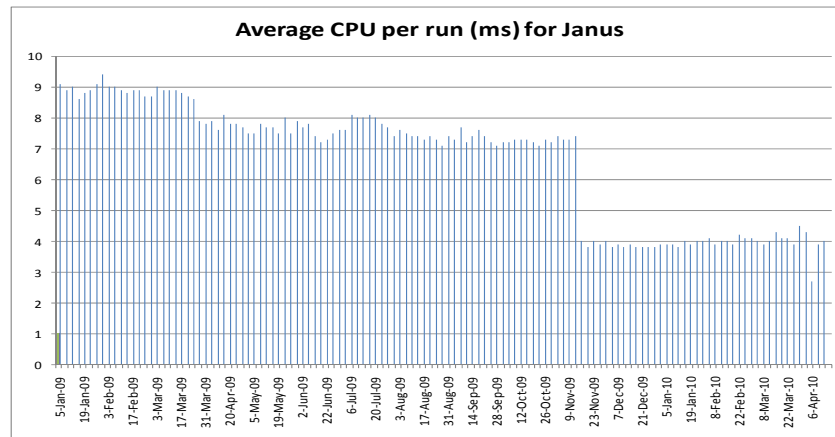
PERA is prod onlines A, P and NX

SYDA is prod onlines I, K, M and R

We’ve got a pile of StoTek ne Sun ne Oracle disk - V2Xs. Mostly because of their support for multiple snapshot copies.

All production onlines have 'read-only' copies on another lpar, in same physical site – due to dasd considerations for using snapshot copies. Also have cross site backups and dual journaling using PPRC for DR.

While we're talking z10's



This from our 'G' environment – one of the biggest. Pick when this online was migrated to z10 machine? And, no it's not a trick question.

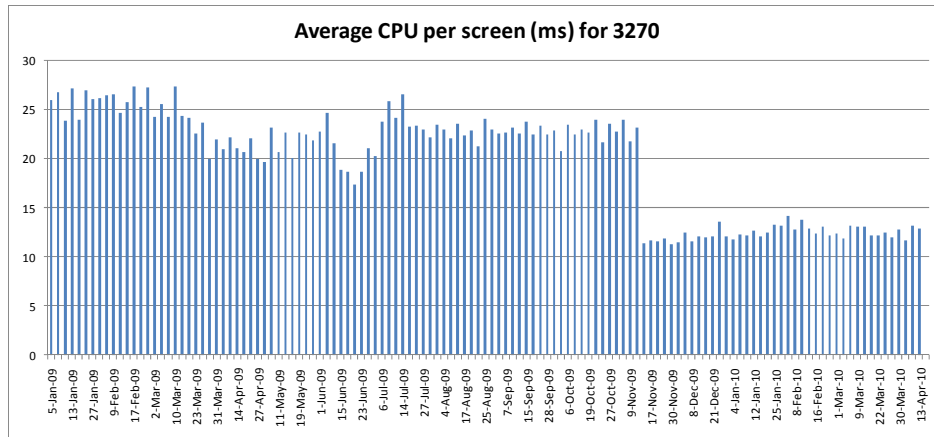
BTW: M204 loves z10's over z9s – Gartner says 1.5 times more powerful, in our experience M204 returns about 1.9 times improvement (consistently seen both during offsite testing and thru onsite experience).

Janus being, of course, the Sirius Janus interface. We've taken to using the word JANUS as the collective noun all of our applications running over a Janus connection of one form or another. We make heavy use of soap over http for browser based work flows (which incidentally, started of as a side project but rapidly exceeded official J2EE direction; and users love it), also email, local DNS resolver, various socket connections to external agencies.

Becomes a little complex when the SLA or Help desk people, etc start referring to Janus as a stand alone application in it's own right. As in "what's the problem with Janus and when will it be back up" as opposed to "we're experiencing connectivity issues to the 'G' environment". I guess that's reflection of how successful the Janus based applications take-up is. For most internal customers now, Janus **IS** Model 204.

Figures taken from Mondays and Tuesdays only – which are our peak days and have more consistent usage patterns. Demand trails down for the rest of the week and figures become correspondingly more noisy (one large transaction can have an out or proportion impact on simply averaging as display here)

pick the z10 install date (cont)



And the same performance improvement occurred for regular (old fashioned) 3270, ie green-screen, traffic. As you would expect.

M204 Usage at a Glance

- 12 production environments (+ 5 Dev, 7 Test, 3 Training, 7 SAF, 3 PST, 2 TPNS and 7 DM1)
- M204 V7R2 + Sirmods 7.7 + most extensions
- ~ 22,000 Concurrent users
- ~ 1100 screens per second
- ~ 24 million screens per business day
- ~ 33 million lines of M204 code, ~ 3.4 million lines of DDL, and ~ 250 thousand lines reference data
- ~ 450 files per online
- ~7.2 Billion mainframe transactions pa
- 0.2 - 0.3 second mainframe response



Being a Sirius Conference, I expect you're more interested in our use of M204. So, staying with the "at a glance" theme . . .

We are, of course, a M204 shop. Running 12 (or 13 or 14 depending on just how / what you want to count) production onlines in a federated fashion. With another as National (central) Index (aka NX) , and one more for a Web Services (aka WS) front end interface.

Been running on M204 V7R2 with matching Sirmods 7.7 since start of April. Cumulation of some 1 and 1/3 years elapsed effort involving Sirius, CCA and ourselves.

Each production environment is implemented as two copies – the actual working, updateable production copy (aka US1) and a redundant, read-only duplicate (aka US2), providing some load balancing and DR capabilities. US2's are refreshed from US1s during overnight batch window but only designed for lookup (without updating) enquiries. As the push into 24*7 availability continues, I imagine this will need to be revisited.

Files per online decreased from ~550, in 2008, to ~450 currently due to increasing take up of large file support. Most files still consist of multiple physical datasets - roughly 1150 M204 file related DD statements overall (per online). 3270 terminals being phased out. Replaced with browser based workflows, via Janus SOAP over http internally; and J2EE externally. Also have MQ and raw sockets (both Sirius and CCA flavours) and horizon and . .

We do about 20 million 3270 transactions a day, with another 10 million from thru our browser based workflow applications via Janus. That's roughly ~38% growth from 2008 (5.2 billion pa).

M204 Usage at a Glance

- 12 production environments (+ 5 Dev, 7 Test, 3 Training, 7 SAF, 3 PST, 2 TPNS and 7 DM1)
- M204 V7R2 + Sirmods 7.7 + most extensions
- ~ 22,000 Concurrent users
- ~ 1100 screens per second
- ~ 24 million screens per business day
- ~ 33 million lines of M204 code, ~ 3.4 million lines of DDL, and ~ 250 thousand lines reference data
- ~ 450 files per online
- ~7.2 Billion mainframe transactions pa
- 0.2 - 0.3 second mainframe response



The National Index + Web Server onlines have true redundant copies to avoid single point of failure issues. Current implementation requires coordinated stop/start switchover sequencing to do backups and refreshes while ensuring one of each pair always available for 24*7 reasons. Requires some client side “smarts”, ie in gateway and prod’n onlines etc to handle the “failover” and recovery.

Only one copy of each of the development and testing onlines. But tight integration with local Centrelink Data-Dictionary, and, therefore, version control chain, enables each Dev/Test online to support multiple application versions simultaneously. Usually at least 3 or more. And a NX and a WS, all talking to each other just like production (obvious blocks/stubs in place to prevent actually talking to the “real” world.).

SAF (System Assurance Facility, think . . . applications pre-prod regression testing) has it’s own lpar- CBRD. Supporting 4 pre-prod onlines (4 x US1 and 4 x US2), a NX (1 and 2) and WS (1 and 2) also. Some things – ie letter writing, have to be deliberately disabled as obviously you don’t want to issue a letter before official release date. Likewise connectivity thru firewall / gateway to external agencies needs to be carefully organised for similar reasons.

PST (Performance & Stress Testing, think) on a different lpar again – CBRH. 3 pre-prod onlines (US1 only) 1 x NX, 1 x WS and 1 x DM1. Mostly used by the J2EE side of the world to provide backend for their testing. Somewhat amusing how it illustrates issues of scale and perspective. We keep getting emails, from the PST people, saying this test or that test of some feature of other, with xxx users (usually small by our standards but big by theirs), requires M204 backend to be active. And later they ask “how was the performance curve”. To which we reply “onlines didn’t even notice the workload, let alone raise a sweat” which always seems to surprise & disappoint them somewhat.

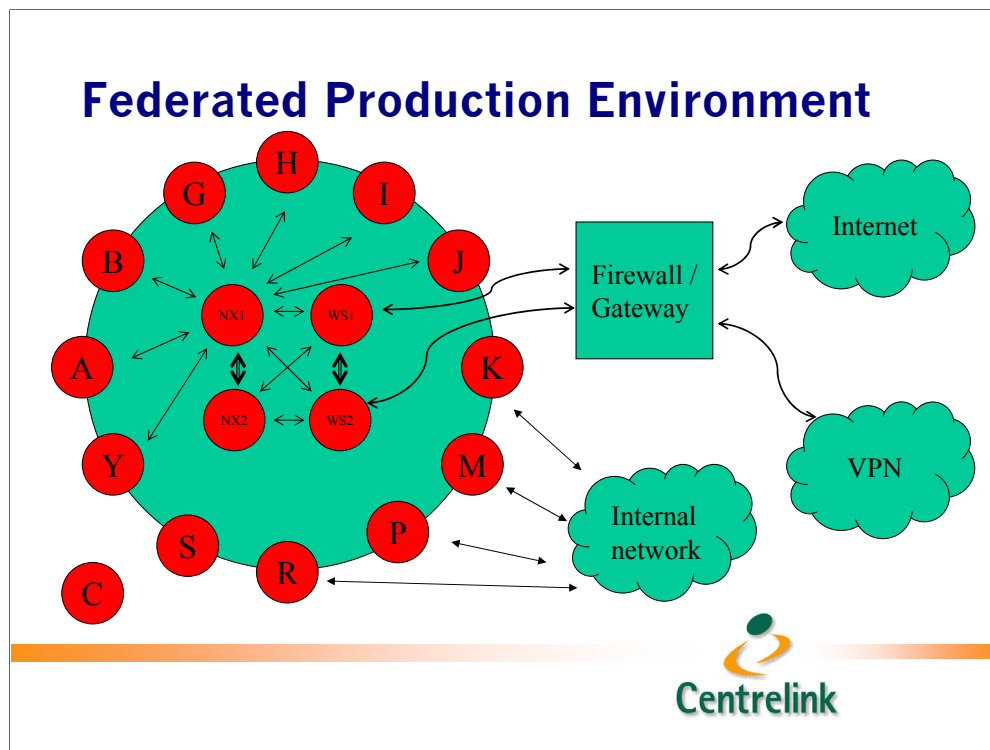
M204 Usage at a Glance

- 12 production environments (+ 5 Dev, 7 Test, 3 Training, 7 SAF, 3 PST, 2 TPNS and 7 DM1)
- M204 V7R2 + Sirmods 7.7 + most extensions
- ~ 22,000 Concurrent users
- ~ 1100 screens per second
- ~ 24 million screens per business day
- ~ 33 million lines of M204 code, ~ 3.4 million lines of DDL, and ~ 250 thousand lines reference data
- ~ 450 files per online
- ~7.2 Billion mainframe transactions pa
- 0.2 - 0.3 second mainframe response



TPNS (IBM's Tele Processing Network Simulator product) on another dedicated lpar. A repeatable environment suitable for M204 nucleus regression testing. I'll go more into that later

BTW : It's not valid to compare lines of code with that from previous presentations. Method of counting has changed – more files are included (5 now vs. 1 previously) ie: infrastructure, Janus applications, DBA tools as well as previous business logic . Most of this code is actually include members so would be recombined many times, in various combinations to make up the actual executables. Ie: even bigger. Reminds me to badger Sirius for smarter compiler to detect/omitted non-executable code.



All production onlines same – just contain different subset of customers along broadly geographical divisions.

Layout and naming conventions set years ago (like, so last century) when M204 mp/scalability constraints and IBM individual cpu capacity limited practical online size. Not an issue any more but general reluctance to re-combine. Shouldn't be. We've tested onlines with up to 11 cpus running flat out. Difficult to arrange resources (buddy, can you spare a lapr & a disk farm) and we run out of test scripts awfully fast - but otherwise satisfactory

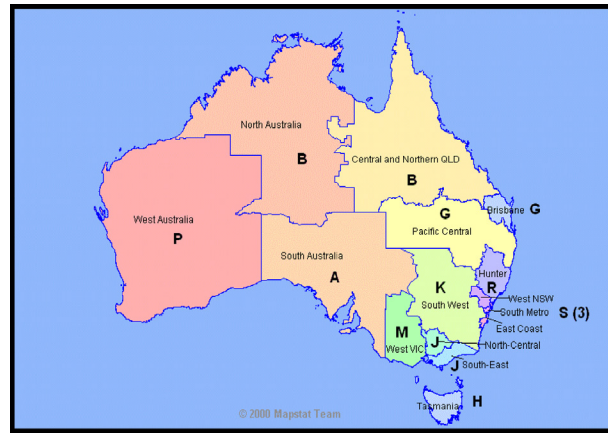
All production environments talk to NX1 and each other. Although “normal” pattern is to mostly stay within the one online. Not all links illustrated as diagram got really ugly really fast

NX1 and NX2 to provide a centralised name lookup/indexing service 24x7. One always up – coordinated stop/start sequencing to do backups and refreshes etc. Not fully transparent yet. NX1 is still (always) primary. Applications need “smarts” to know how to switch between them. Not all applications fully support this yet.

WS1 and WS2 to provide web server front end(s) 24 x 7. Customers can access their data 24x7 via Internet. Also external agencies can to various lookups, checks etc over VPN connections. Like NXs - one always up, coordinated stop/start sequencing required to do backups and refreshes etc. Also not fully transparent, requires application & gateway “smarts” to support switching. And again, not everything does yet. Also has smarts to hand-off requests to data onlines once initial location/authentication done to reduce latency.

“C” on it's own as although production online, doesn't actually have customer data. Treated same as others from operational viewpoint. Can talk to others etc. Always talk of making better use of it to support non-customer data. Think child care centres.

Centrelink Onlines



Just showing the physical coverage afforded by the various production onlines.

Names and coverage have some degree of historical relevance, reflecting when data centres were physically located in the state they served. Ie B for Brisbane (we'll ignore the fact it does north Australia and doesn't include Brisbane city anymore), S for Sydney, M for Melbourne etc. Then others as processing growth requirements required splitting onlines. Ie R after S. But couldn't carry one meaningful names. Ie G after B

Now scalability is such we don't expect to split for a while. If at all.

How Busy is Busy?

- Average ~3.44 billion \$function / object calls in a 1.5 hour TPNS (regression) run. ~47% Sirius, ~26% CCA/Rocket; and ~26% ours
- ~ 512 million (~14.8%) and climbing are various object related calls
- top five are \$LISTINF, \$ONEOF, \$DSSNUL, \$LISTINFI and \$DEBLANK @ ~194, ~169, ~162, ~159 and 148 millions respectively.
- and then object call (\$\$CLASS_METHOD) at ~145 million (~0.4%)



An online not consuming a whole cpu or more is generally considered not worthy of interest. Let alone busy. 2.5 to 4.5 (z10 cpus) is normal.

About a 6.8% increase in \$function calls since 2008 presentation (was ~3.22 billion). But the really interesting point is the growth in our use of Sirius objects, climbing from ~12.9 million (~0.4%) in 2008 to what you see here (~512 million is ~14.8%, umm, what's that? some ~37~39 times as much – interesting that matches the growth in transaction numbers too). Reflects the changing demographics (training, experience) of average Centrelink application coder.

The top five in 2008 were pretty much the same at \$ONEOF, \$DSSTBL, \$LISTINF, \$LISTINFI and \$DSSNUL; at ~207, ~190, ~184, ~169 and ~165 million respectively. If I'm subtle enough about, maybe I can get Sirius to look at (further) optimising \$LISTINF[I] for a performance gain.

\$function counting done by a local hook zapped into the \$function dispatch table - KZFNDTAB, and reported via associated code in official CUSTTERM exit. I zapped my way in originally and had a lot of fun and games by switching \$functions under the covers (and err, making mistakes). You know, call \$x but got \$y kinda thing, kept me amused for minutes at a time - others didn't appreciate it thou. Makes it difficult to start an online, let alone do anything worthwhile. Function counting only activated in TPNS, not anywhere else (Dev, Test or Prod etc). In theory, it should work, but we've never needed the information to be that current to justify enabling it. Consequently, so far, my curiosity has been over-ruled. Which is really quite fair enough, given the circumstances.

Online Peak load

	#users	scr sec	cpu%	Janus cpu%	3270 %cpu	DKIO/ sec	numbufg	maxbuf	#servers	
A	1016.00	101.70	214.54	53.34	70.71	4912.00	100000	80000	205	12:15
B	2738.00	127.30	307.57	103.96	99.34	8733.00	100000	80000	205	12:30
C	34.00	0.63	1.58	0.06	0.65	53.00		5000	10	16:00
G	2715.00	134.50	314.81	90.06	122.79	8102.00	100000	78000	220	12:30
H	751.00	38.74	152.39	38.94	64.99	4257.00	0	20000	50	14:30
J	2655.00	128.30	471.15	145.75	165.21	7096.00	100000	80000	220	14:30
K	2243.00	110.00	457.08	119.02	185.98	5731.00	100000	80000	205	12:00
M	2236.00	112.50	405.53	129.82	137.10	5941.00	100000	80000	205	11:45
P	2138.00	102.50	225.85	70.02	73.41	5844.00	100000	80000	205	12:45
R	1901.00	101.30	434.87	140.35	144.16	5356.00	100000	80000	205	11:45
S	2896.00	145.00	504.89	155.27	181.21	8016.00	100000	80000	220	11:45
Y	29.00	7.86	5.67	0.00	0.03	3.06	0	40000	25	11:30
total	21352.00	1110.33								
L WS1	8.00	12.96	7.22	0.00	0.00	276.20	0	1000	100	17:00
NX1	45.00	70.72	25.63	0.00	0.00	1165.00	100000	80000	100	2:45



A few basic statistics show peak online performance. 15 minute intervals, taken from Tuesday 30th March – a two weekly cycle peak.

Centrelink has a two weekly cycle (matching most payment cycles). Consequently every 2nd Tuesday is our peak day, with the immediately previous Monday coming a close 2nd and then a regular trail-off thru the rest of the week. Very quiet on Fridays.

Online Day :

02:00am - 01:00am (next day, 23 hrs); Mon - Fri,

04:00am - 08:00pm; Sat

04:00am - 22:00 (10pm); Sun

V7R2 at a glance

- Current Centrelink production version + Sirmods 7.7 (although not exploiting new features yet -> June '10)
- Perhaps a ~0.7% performance improvement over V7R1 if we look at carefully (definitely no loss)
- Native support for Repeating Field Groups (RFGs)
- Exploiting RFGs requires new file type FILEORG=X'100', otherwise drop in replacement
- Switch to RFGs returns ~8% performance improvement (seen under TPNS with only 13 major files reorganised)



If you've been around M204 for any time you've probably come across the M204 feature of Repeating Fields, which Centrelink then groups together making (logical) Repeating Groups. However, this requires a certain degree of care on the part of the coder to keep fields making up the "groups" in the correct sequence. We have special underlying api's, called Data Modifiers, used to extract and store data in M204 records while retaining the correct "group" layout. Which, in turn, sees much use of specialised functions to convert null fields into a single underscore and back in order to retain the overall consistency (see \$DSSNULL on most heavily hit list previously).

M204 V7R2 actually grew out of an initial CCA proposal in the high availability arena. Their initial proposal was for a sysplex aware version of M204 with full high availability support in all varieties. Ie: 24*7, disaster recovery, transparent fail over, etc. Would have essentially been built around a M204 native PPRC-like implementation, so M204 could transparently switch physical databases as required. Some nice points, in a fluffly, hand waving kinda way, but scared the stuffing out of us (we have enough challenges with real IBM PPRC).

So, a revised proposal for native support for RFGs was developed, addressing the significant roadblock in our 24*7 direction, ie: file reorganisation during the batch window. Like everybody, as we're being pushed further in the 24*7 direction, the overnight / weekend batch window is rapidly contracting. Most BATCH204 work already converted to BATCH2. Biggest issue now with time required for file reorganisations whether routine (ie cleanup dirty deletes), or the odd emergency (run out of space etc). But, most noticeably, when production release requires adding a new field to an existing repeating "group" to retain structure and sequencing. By M204 now "knowing" the structure of a RFG, much overhead related to managing them disappears (well, from our viewpoint, Alex might disagree). Simple things like being able to assign a default value to a repeating field make life significantly easier.

V7R2 at a glance

- Current Centrelink production version + Sirmods 7.7 (although not exploiting new features yet -> June '10)
- Perhaps a ~0.7% performance improvement over V7R1 if we look at carefully (definitely no loss)
- Native support for Repeating Field Groups (RFGs)
- Exploiting RFGs requires new file type FILEORG=X'100', otherwise drop in replacement
- Switch to RFGs returns ~8% performance improvement (seen under TPNS with only 13 major files reorganised)



M204 V7R2 is different enough that it requires Sirmods 7.7; the details of which I'll leave to Sirius. Suffice to say that, in the usual Sirius fashion, it does everything the prior versions do plus more.

So far, it's all proving very stable.

BTW : Journals/jlogs just similar yet different enough to be dangerous. There's an increased header size and some changes to actual logged updates; but MERGEJ, AUDIT 204 (and our utilities – FASTMON, MRGJRN) “cope” (well, don't explode) with that. But, obviously a V7R1 journal not good for V7R2 recovery and vice-versa. And a V7R1 merged with V7R2 journal not much good for anything. We managed to do all that, and more (particularly when alternating V7R2 and V7R1 in some of our development onlines), which caused some dramas for some post-processing jobs which walk journals/jlogs for analysis, security checks etc.

V7R2 Field Groups

```
- DEFINE FIELDGROUP FieldGroupName  
  
- DEFINE FIELD FieldName  
  WITH FIELDGROUP FieldGroupName [AND  
  [(attribute1 attribute2 ...)]]  
  
- FEO FIELDGROUP FieldGroupName  
  ...  
  END FOR
```



These are just a few simple examples defining a M204 Repeating Field Group, that I've copied from CCA's "M204 Release Notes" manual. All pretty much as you'd expect.

There's also a number of new field attributes and some specialised auto-magic fields (eg create-time, create-user etc). Far more than I can hope to get thru here without basically just regurgitating the manual and running overtime. And I'm sure I'm boring you enough already

TPNS introduction

- IBM product : Tele-Processing Network Simulator
- Capture and replay facility
(for 3270, and TCP/IP and others)
- Captured traces converted to scripts
- Can be replayed as desired
(via Control-M scheduler package)
- Also captures result screens. Used for comparison during replay to determine correctness



TPNS is an IBM product originally designed to capture and replay 3270 traffic, comparing the results with what originally occurred. Has been extended to support TCP/IP and pretty much anything else that can be scripted.

We normally take a new capture after every major release (quarterly) – after any application issues get resolved. Requires copying (via snapshot, then real I/O) of all M204 database files before onlines (one prod, matching NX and DM1) come up. And then capturing all 3270 traffic (requests and responses) against them during the day. Also need to capture online JCL, parameters and any BATCH2 jobs run against it, etc. Over the years a checklist of manual steps and jobs has evolved to capture everything required. Janus traffic is captured by means of TCPLOG option on the JANUS port DEFINE command(s) –writing to a g00v00 dataset.

Probably a day and a half worth of subsequent effort to massage all captured information (jcl, parm, vtam 3270 trace, Janus tcplog data, batch2 jobs) into form (scripts) suitable for replaying thru TPNS. For example, clean up/delete those scripts that are communicating with other onlines. Using a set of home grown SAS jobs to do necessary processing.

Replay can be initiated manually or usually highly automated and driven thru scheduler package. Centrelink uses Control-M from BMC for this. Brings up onlines, submits BATCH2, 3270, Janus workload. When complete, brings down onlines, restores databases to original state from backups, runs statistical collection and analysis jobs; ready for next time. When all goes well, can get 2 or 3 runs thru overnight without intervention. “well” from a Control-M perspective can include runs with snaps, or coming down ugly. The cleanup is pretty good. But when it goes bad it can leave quite a mess – this job overlays that one, something didn’t run whatever.

TPNS introduction

- IBM product : Tele-Processing Network Simulator
- Capture and replay facility
(for 3270, and TCP/IP and others)
- Captured traces converted to scripts
- Can be replayed as desired
(via Control-M scheduler package)
- Also captures result screens. Used for comparison during replay to determine correctness



Can speed up, slow down traffic and change the mix – set up particular test scenarios as necessary.

Due to timing and coordination issues with others jobs, not exactly 100% repeatable, but within a known margin (~1 ~ 2%). Load balancing between the lpar can also bite, as can obscure hardware issues (ie TR instruction on one lpar flushing cache in another - resolved in z10s but enough of a drama to force us to code asm loops rather use TR). Typical run is one & half hours elapsed to simulate most of the regular business day (obviously some time compression occurs).

Using TPNS

- Fully functional M204 online, complete with “real” user work load
- Can test : load module changes, hardware changes, application changes (within known limitations), destructive testing, recovery testing.
- Portable
- Our night shift is Sirius daytime
- Found many problems
- Collect statistics for analysis / comparison / history



TPNS provides us with fully functional onlines, identical to a production; complete with “real” users doing “real” things (just not). There’s 3270 traffic, Janus traffic, MQ and Batch2 – all exactly as they really happened (although usually time compressed). Ideal regression tester for M204 nucleus changes.

Provides a flexible environment suitable for regression testing of load module & other changes, destructive testing and recovery testing. Can handle UL code to a lesser extent – as long “user” behaviour (as displayed on screens) is unaltered from what was recorded else noted as “error”. So not generally suitable for new UL (see SAF) but can handle infrastructure UL – common components, api’s etc. But is suitable for performance comparisons – this algorithm vs. that one (given they don’t change user behaviour). Also suitable for ad-hoc testing, investigation and reproduction of errors. Run it up, start the load and then do whatever needs to be done. Ie: take snap, etc

Since it’s on it’s own lpar, the entire lpar can be picked up and moved around to facilitate hardware testing. Recently done to compare z10s vs. z9s – at IBM’s test laboratory in France. Technical trivial, but politically very sensitive and complex operation – you can imagine the issues involved in taking extensive customer information out of the country. Ended up doing the full 007 thing with diplomatic bags, and cartridge tapes handcuffed to couriers.

Basic usage is simple, prepare whatever is to be tested – new load module, different online parameters, different file organisation etc. Start TPNS typically using scheduler. Sit back and wait for results to come in. Collect, collate and analyse

Using TPNS

- Fully functional M204 online, complete with “real” user work load
- Can test : load module changes, hardware changes, application changes (within known limitations), destructive testing, recovery testing.
- Portable
- Our night shift is Sirius daytime
- Found many problems
- Collect statistics for analysis / comparison / history



TPNS has found many problems before they become problems. For instance, as I was writing this, the overnight runs are showing a change in the M204 error message counts. Tracking that back (bit painful, running audits etc) and knowing what we'd installed, points us to a zap which is not playing nice with others. Saves like that are easy and undramatic via TPNS, with the “problem” getting stopped & fixed before it gets anywhere near to production

Post processing includes SIRTUNE and SIRAUD reports as well as local jobs for analysis of top 50 procs and files; cpu consumption, average user load etc etc. A single run of TPNS currently triggers collection and logging of some 85 statistics for comparisons (pew – good job for new starters). Including the simple like run date & time, build identification. The obvious such as elapsed, cpu and srb time; EXCPs, journal and jlog blocks, \$function calls; Janus traffic - reads and writes; daemons in use. The more telling - average number of concurrent users, screens and procs executed; %pcpu. TPNS also tells us how many scripts have different results from when they were recorded. And then ... we collect and collate error counts; ie how many of each the M204, MSIR and USER prefixed messages. All those statistics are normally pretty consistent (within a small margin (+/- ~1%) so variations can be seen fairly easily (understanding the cause is another issue).

I was going to include a line of two from one of our spreadsheets here – but it just doesn't fit. It either wraps and wraps and warps, or is going to extend out to about there. Either way, it's pretty much unreadable. I've got a copy if you're really interested to have a look. Come and see me later.

Got a significant workout as part of the V7R2 RFG project.

TPNS and Janus traffic

- TPNS not ideally suited to Janus
- Our first implementation read the audit trail to reconstruct the traffic. Slow and hard.
- Now logged by TCPLOG option on JANUS port DEFINE
- Replayed using Sirius supplied replay tool
- Still issues with expiring cookies, tokens and application level security.
- Collect JANUS TSTATUS, JANUS STATUS and JANUS LIMITS statistics



While TPNS is TCP/IP aware. It's trace / replay facilities orientated towards hand scripted TCP/IP traffic – not ideally compatible with trace / replay of Janus traffic as we'd prefer.

Originally we would use JANUS TRACE 31 to write everything to the audit trail and then reconstruct it from there. Required an IBM TPNS & TCP/IP expert (rare) to write that for us. Ok, but, problem was it took over best part of two weeks elapsed processing to run to convert audit trail into scripts. Many jobs ran for 3 to 4 days each. Required multi-volume PDS-Es and large files on dasd and multi-tape intermediate datasets. Prone to silly errors (particularly space or bad tape) requiring job restart.

Subsequently, Sirius developed the TCPLOG option to log all JANUS traffic across a particular port to a flat file. They (hi, Alex) also developed an application to read that file and replay the traffic for us. All we had to do was stitch Alex's code into our TPNS schedule.

TPNS replay of Janus traffic now functions correctly. TCP/IP arrives at the target online which does, well, whatever it supposed to do based upon the Janus definitions (which we also captured when we captured everything else). M204 apsy'es get called, UL code is executed and so on. With appropriate results being sent back, again via Janus, to the originator. All good.

We've got some internal issues with expired cookies, tokens and/or applications level security all working together as designed; preventing some applications or subset thereof, from running. That is, they run from a TPNS replay of Janus traffic viewpoint, but the application itself rejects in depth processing – for a variety of reasons. So we're not exercising the applications UL side in sufficient depth to be comfortable (important as Janus take up is now on tipping point of exceeding 3270). But it's correct / functional from the viewpoint of TPSN rerunning Janus traffic

TPNS and Janus traffic

- TPNS not ideally suited to Janus
- Our first implementation read the audit trail to reconstruct the traffic. Slow and hard.
- Now logged by TCPLOG option on JANUS port DEFINE
- Replayed using Sirius supplied replay tool
- Still issues with expiring cookies, tokens and application level security.
- Collect JANUS TSTATUS, JANUS STATUS and JANUS LIMITS statistics



Issue is well understood, we have a task to tweak applications code so that this issues are bypassed but only under TPNS. We're using "Luche globals", ie `$SETG_SYS('\SYS.TPNS', '')`, to modify UL code to be more accommodating under TPNS. Funny problem in that applications coders don't want / understand those changes and they tend to remove them; meanwhile new code doesn't include them. Ongoing drama in keeping things in sync. Another approach, so far only discussed internally, is to extend Alex's code to delve into the data being replayed, modifying it as appropriate on the fly. Fabulous technical challenge (especially for me as an ex-firewall jockie).

Have some UL to collect JANUS TSTATUS, JANUS STATS and JANUS LIMITS statistics during EOJ processing. Almost good enough, but get's lost if only comes down with a thump. Must have another try at convincing Sirius to supply necessary internal control block(s) details so that I can display from CUSTERM exit instead. But understand why they're reluctant.

TPNS testing of V7R2 alphas/betas

- Started late Dec. '08. Took about 15 months
- Our tracking spreadsheet has 29 pages, with 567 entries, and doesn't include things found/fixed by CCA + Sirius.
- 313 refreshes of entire M204 + Sirmods object code base
- Over 1030 runs of TPNS test suite
- Often required code changes and / or file re-org first to take advantage of new features or fixes.
- Significant effort involving dba's, file designers, architects and applications coders.



1030 runs of TPNS doesn't really sound like a lot if you say it fast. But, when you do the sums, that's ~2 and a ¼ a day. Every day – weekdays, weekends and public holidays (and we're not quite that dedicated). At 2 and ½ hours elapsed for each run. Plus a bit of turn around time – preparing new load modules, re-org'ing files, changing UL code, etc; and (of course) collecting and reporting bugs.

And that, is all in addition to our normal use of TPNS, for current production version fixes, changes etc – about 710 more runs over the same period (so average now ~3.8 a day). You can see how it became quite a job. Poor us. ☺

We normally have operators kick-off a run or two overnight, weeknights. Which is fine, when they run, but makes it awful when we've got to pick up the pieces next morning if they don't. The really good thing about this method, is that Sirius could VPN into us and have a look (since it was daytime for them). Sometimes we'd have the next fix before we even knew it was broken.

The most annoying thing found

- When CCA fixed the compiler to correctly enforce 8 character group name limit. Ie:

```
IN GROUP <name> MEMBER <blah> . . .
```

- Appears we relied on existing broken behaviour a bit too much (9 character group names).
- Unfortunately, we all (CCA, Sirius + Centrelink) spent a week trying to work out what we'd done in our code, or find the file we'd incorrectly reorganised, etc.



It turns out we build that technically invalid 9 character group names, in some (wouldn't just be one now would it?) of our skeletons (which aren't the same as macros, api's or include code – full fledged home grown code generator, but that's another story). Previously M204 would silently truncate that to 8 characters but, with the fix, quite correctly raised it as a syntax error at compilation time .

But . . . We use those skeletons as input to our code generators – so it's been generated into squillions (that's a technical term) of bit's of code all over the place. Thousands and thousands of times. Arrgh. We started trying to swat them one by one in TPNS. Bit like playing wack-a-mole. For every one we “fixed” and now compiled, another handful appeared (as the compilations moved on).

So, CCA removed the “fix” for us. We're supposed to fix our code so they can reinstall it. I wouldn't recommend they hold their breadth. Which, I guess means, they'll fix it again before we're ready. Foot, meet bullet.

The most surprising bug missed.

```
begin
  %i      fixed
  for %i from -50 to 0
    print %i
  end for
end
```



It's not a trick question, so I'll let you study that for a while, to work out what's wrong. Anybody? No. I'll give you a hint, there's no output.

Lie I said, it's not a trick question, there should be output.

Ok, being a Sirius conference that should have been

The most surprising bug missed.

```
begin
  %i      fixed
  for %i from -50 to 0
    printtext {~} {%i}
  end for
end
```



printtext {~} {%i}. Or similar. But none the less . . .

Interestingly, changing the definition of %i to float worked just fine.

Luckily this got caught by a cluey developer early in our staged rollout to production, and they were good enough to notify us. And we to CCA + Sirius. But it put the rollout back a week. We originally had it timed, just so, for when Gary, Dave and John were onsite; and Alex on the hot phone. How's that for vendor support? But, you know what they say about the best laid plans . . .

Mind you, that's still way better than the alternative of finding it in production while it did who knows what damage.

It just goes to show how trickie testing is. Everybody and everything missed it. Although, I guess it could be argued that the developer who'd optimised that bit of code (obviously just a little bit too much) should have had a test for it. Who knows, maybe they did.

Sirmods zaps

- 400+ developers, various levels of skills, interest and ability
- 5 Dev, 7 Test onlines, 4 SAF, 3 PST, 2 TPNS and a few more
- ~15 concurrent versions / releases
- And Production, for a serious workload



Every wondered where all those zaps to a Sirmods version come from? Well, we can't claim credit for all of them but . . . Well . . . Shucks . . . It's not like we don't try.

Seriously, we've got some 400 plus developers pounding away, full time doing all manner of weird and wonderful things. Trying everything from the manuals, and more. If there's an issue lurking out there, there's a good chance they'll find it. Problem is they mostly don't report it – just ignore and try again. So, I've got a bunch of jobs that scurry around every morning (well, when I logon) looking for any snaps, anywhere. Unfortunately, by then, the journals/jlogs have merged to tape and it's fairly tedious getting the necessary details for the bug report.

Mind you, they're supposed to be developing their own UL code for whatever business application requires. Anything else that pops up is just a bonus. As long as it occurs anywhere but production – we don't mind.

Even when the code runs safely everywhere, passing all known and applicable regression tests. Putting it to production is another issue. Suddenly instead of one developer, or a handful of testers; there's a couple of thousand users relying on it, possibly truly in parallel (we're MP with multi-server), which tends to shake out concurrency and timing issues.

I feel for Sirius, I really do.

Recovery / regenerate testing

- What about those rare “once in a blue moon” bugs ?
- We’re lucky to have a seriously gifted team member
- Many complicated test scenarios designed around strategic failures involving timing, changes and / or intervention.
- Focused on Recovery/Regenerate but equally skilled with all M204 and local utilities.



The one thing about a database is that it doesn't lose data. If Centrelink loses data, that means customers don't get paid. Everybody involved takes that very seriously. It's one thing to have to explain bouncing a online in the middle of the day, it's a whole different world of hurt when data gets lost at the same time. So we don't like recovery / regenerated problems.

Luckily we've got a seriously gifted team member when it comes to testing regenerate / recovery. I call him "Touch of Death". It's hard to explain, he's just got the "gift", the magic fingers, the art, the skill and dedication. Something that, in comparison, the rest of us lack. Mind you, "genius tester" is not his official job description – he's just so good at it that our team passes all our recovery/regenerate testing over to him. It's almost a cliché for us now, Rod's not working until he's broken recovery/regenerate at least once. What's more, one is not really trying – two or more is the target.

For example, Rod's got a job to regenerate the biggest and nastiest files from our biggest online once a week, every week – just in case. And you know what – he did find something. Long since fixed now, but the job still runs, just in case it comes back, or something new appears. He has an entire suite of jobs to construct recovery / regenerate edge case scenarios. For example: image an online failure, followed by roll back failure, followed by another rollback failure, followed by roll back ok but roll forward failure, followed by roll back & forward ok, then more real updates, then another online failure necessitating a regenerate thru them all. And yep – that found something to.

Recovery / regenerate testing

- What about those rare “once in a blue moon” bugs ?
- We’re lucky to have a seriously gifted team member
- Many complicated test scenarios designed around strategic failures involving timing, changes and / or intervention.
- Focused on Recovery/Regenerate but equally skilled with all M204 and local utilities.



There's many combinations and permutations of such jobs to making up our recovery test suite. Too many to list them all. However, in general, they all involve breaking recovery / regenerate some how, somewhere and then seeing what happens next. But the catch is they have to be consistently repeatable. It's all very well to cancel a recovery and delete a few lines from the journal but that proves nothing. So we've jobs with undersize or different sized checkpoints and / or journals, tests where a checkpoint or journal no longer responds. Some are volume based, some are silly edge cases, some have strange / extreme combinations (as above). Some require consistent manual intervention – ie cancel the online x seconds after a successful sub-trans checkpoint, or after the last update but before the M204.0844 RECOVERY IS NOW COMPLETE message.

If you're particularly interested, I've got a copy of our test plan which you can have a look at – although, I don't have the actual job's.

End of presentation

Questions?



Well, that's quite enough of me.

Thanks for your attention. I've got copies of various spread sheets if you're particularly interested – see me after.

Any questions?