FIO

CERN IT Department

Service Incident Reports

WLCG collaborations workshop

Olof Bärring CERN IT/FIO

CERN - IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

22/3/2009

Summary

- Incidents
  - How do know you have one?
  - How can end-users help
- Service Incident Reports
  - When?
  - Why?
  - How?
- The most common cause for incidents
  - Managing changes

Incidents

- How do you know you have one?

  An 'Incident' is any event which is not part of the standard operation of the service and which causes,**or may cause**, an interruption or a reduction of the quality of the service.

- … assume it will!

CERN - IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Incidents

"The objective of Incident Management is to restore normal operations as quickly as possible with the least possible impact on either the business or the user, at a cost-effective price"

- … true but first:
  - **Tell everybody even if you don't know what exactly is the problem or its cause**
    - **If other services depend on yours tell them as well**
    - **Who announces to the users?**
    - **Don't put names of people (e.g. service mgrs) in your announcement**
      - **Strip off mail headers**
    - **Clear and concise, end-user focused:**
      - **'Router lcg234.cern.ch crashed'**
      - **'Service XYZ is experiencing some problems'**
  - **Update (append!) to announcements as restoration progress**

How can users help

- Look for announcement before asking
  - If we do our job ok and actually managed to put out an announcement…
- Forward announcements to your internal forums
- .. Be patient
  - Avoid contacting service mgrs directly

Summary

- Incidents
  - How do know you have one?
  - Lifecycle
  - How can end-users help
- Service Incident Reports
  - When?
  - Why?
  - How?
- The most common cause for incidents
  - Managing changes

Service Incident Reports: When?

- Degradation goes beyond some MoU target for any service classified as critical for at least one of the VOs !
- SCOD asks for it !
- When it's useful for your own purpose
  - Tracking of incidents and the restoration → your knowledgebase for ***when*** it happens next time

Service Incident Reports: Why?

- Any noticeable service outage deserves an explanation
- Choices
  - Wait for and answer the questions when they come
    - Even a well-explained event will be distorted down the line as the information is spread
    - Your mailbox is the knowledgebase
  - Upfront detailed explanation somewhere where everybody can see it
    - Including yourself when you have the same incident in the future

Service Incident Report: How?

- Written report with an appropriate level of details
- Focus on:
  - What went wrong
  - Who was affected (impact)
  - How and when you noticed
  - How and when you announced
  - Main steps of the service restoration
  - When the service was restored and announced as such
  - Follow-ups/actions
- A timeline is useful
- Avoid names or other details of CIs (machines, people, …)
- Be honest
- Attempt to classify the cause
  - Change
  - Bug
  - DB
  - Human
  - The network
- In ITIL terms SIR writing activity probably closer to Problem Management

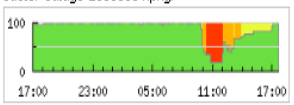
Description

All CASTOR Oracle databases went down at the same time following a 'transparent' intervention on the private network switches between the NAS headnodes and the storage. This caused a general service outage of the stagers and CASTOR name server (as well as other central services).

Impact

All CASTOR and SRM production instances were down for approximately 3 hours:

- castor-outage-20090304.png:



Time line of the incident

- 09:43: oracle databases went down
- 10:01: users started to report problems accessing CASTOR
- 10:26: CASTOR service manager submitted a first incident message for posting at the service status board (http://it-support-servicestatus.web.cern.ch/it-support-servicestatus/IncidentArchive/090304-CASTOR.htm)
- 10:40: Most of the databases are back, the srm-*-db databases are still down for now.
- 11:30: Most databases back except srm-atlas-db and c2cmsdlfdb
- 11:34: CASTOR name server daemons restarted. This was required in order to re-establish the database sessions
- 11:36: service status board updated with the information that most databases were back
- 11:45: **castorcms** recovered
- 11:49: All databases back
- 13:00: **castoratlas** and **castorlhcb** recovered
- 13:21: All SRM servers restarted
- 13:30: **castorpublic** recovered
- 13:29: **repack** restarted
- 16:48: **castorcernt3** recovered

The network bandwidth plots for the various instances (see bottom of this page) gives a good indication of the outage period for the 5 instances.

Analysis

The switch intervention had *not* been announced. There was another intervention on the same switches earlier this week (2nd of March), which had been announced http://it-support-servicestatus.web.cern.ch/it-support-servicestatus/ScheduledInterventionsArchive/090302-IT-DES-DBs-Switches.htm .

The following analysis was provided by the DBA team:

- *the planned intervention for upgrading one of the storage switches to a larger one was made of several steps (power off switch, decable switch, change switch, recable switch and power the switch). While the switch is not powered, the machines use the other side of the Ethernet trunk/bounding*
- *unfortunately the switch has been powered before all was recabled*
- *this has lead (9h43) to have the machines reuse the first part of the trunk/bounding while not all connections were available*
- *this lead to the Oracle clusterware not being able to read the cluster voting files/disks and thus reboot*
- *most of the systems have restarted fine (typically 9h46 OS, typically 9h53 database with the crash recovery), some had to be manually restarted as they had been restarted before the recabling was finished*

When the databases started to come back, the CASTOR2 stager daemons automatically reconnected. This was not sufficient for recovering the service. The CASTOR name servers were stuck in stale ORACLE sessions. That problem was discovered by the CASTOR development team and the servers had to be restarted. However, even after the name servers had been restarted several CASTOR instances (**castoratlas**, **castorlhcb**, **castorpublic** and **castorcernt3**) were still seriously degraded. It is likely that the CASTOR2 stager daemons were stuck in name server client commands. A full restart of all CASTOR2 stager daemons on the affected instances finally recovered the production services by ~13:00. All the SRM daemons were restarted at 13:21 for the same reasons.

The recovery of the less critical instance **castorcernt3** was delayed because its deployment architecture is different. It was only in the late afternoon when it was finally understood that the instance was stuck because it is running an internal CASTOR name server daemon (this will become the standard architecture with the 2.1.8 production deployment). After having restarted the daemon (and the stager daemons) the instance rapidly recovered.

Follow up

- The existing procedure for recovering CASTOR from scratch (PowercutRecovery) needs to be reviewed. The recovery of some of the CASTOR stager instances took longer than necessary. The reason is likely to be that although the database connections had been automatically re-established, most of the threads were stuck in CASTOR name server calls (this cannot be confirmed).
- Next time a message should also be posted to the Service Status Board when the service has been fully recovered.
- The Change and Incident Management processes urgently need to be strengthen and work has started to [illegible] on approach based on best practices

Summary

- Incidents
  - How do know you have one?
  - Lifecycle
  - How can end-users help
- Service Incident Reports
  - When?
  - Why?
  - How?
- The most common cause for incidents
  - Managing changes

To change or not …?

- Is change bad?
  - The dogma for LHC start-up has been stability,
  - "It's working! don't touch it"
  - Truth is that everything changes… **All the time!**
    - Configuration (s/w, h/w): every day
    - Linux updates: every week
    - Linux OS: every ~18-24 month
    - Middle/soft-ware: more often than desirable
- … or is it just the change control that is bad?
  - Assume that change is needed for improving something
    - Functionality for end-users
    - Service operation and stability
- *Managing* changes rather than avoiding them?

When? Deployment strategies

- Baby-steps
  - Trickle of changes one-by-one
  - Each of which may be treated independently
  - If something goes wrong, easy to rollback
- Periodic scheduled
  - Aggregation of changes
  - Freeze, test and certify
  - If something goes wrong, rollback may be difficult
- Big-bang
  - Basically the same as periodic scheduled changes though not necessarily ‘periodic’
  - Accumulate changes for a long period, which may include major upgrades to more than one component

When? Deployment strategies

- Baby-steps
  - Trickle of changes one-by-one
  - Each of which may be treated independently
  - If something goes wrong, easy to rollback
- Periodic scheduled
  - Aggregation of changes
  - Freeze, test and certify
  - If something goes wr
- Big-bang
  - Basically the san
    though not neces
  - Accumulate changes for a long p ich may include major upgrades to more than one component

But... it usually only breaks after a while due to destructive interference of accumulated changes ☹

CERN - IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

When? Deployment strategies

- Baby-steps
  - Trickle of cha
  - Each of whic
  - If something g
- Periodic scheduled
  - Aggregation of changes
  - Freeze, test and certify
  - If something goes wrong, rollback may be difficult
- Big-bang
  - Basically the same as periodic scheduled changes though not necessarily 'periodic'
  - Accumulate changes for a long period, which may include major upgrades to more than one component

Lacks the virtue of establishing change as routine. Between two big-bangs there may be an universe ☹

CERN - IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

When? Deployment strategies

- Baby-steps
  - Trickle of changes one-by-one
  - Each of which ma
  - If something goe
- Periodic scheduled
  - Aggregation of changes
  - Freeze, test and certify
  - If something goes wrong, rollback may be difficult
- Big-bang
  - Basically the same as periodic scheduled changes though not necessarily 'periodic'
  - Accumulate changes for a long period, which may include major upgrades to more than one component

But… the goal for m/w provider should be to allow for revocable updates ☺

CERN - IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

When? Deployment strategies

- Baby-steps
  - Trickle of changes one-by-one
  - Each of which may be treated independently
  - If something goes wrong, easy to rollback
- Periodic scheduled
  - Aggregation of changes
  - Freeze, test and certify
  - If something goes wrong, rollback may be difficult
- Big-bang
  - Basically the same as periodic scheduled changes though not necessarily 'periodic'
  - Accumulate changes for a long period, which may include major upgrades to more than one component

Preparing a change

- Plan
  - Develop a timeline with details who is going to do what and when
- Announce
  - Negotiate the date/time with VOs
- Exercise
  - Try out the plan on the preprod infrastructure
  - Time it! (adjust your announcement if necessary)

- Ulrich: lxbatch

General points

- coordination and handover should be done by e-mail, to keep everyone informed
- CastorUpgrade21612to2173

Preparations

When	What	Who	Status
Thu, May 8	Mail experiment and copy MOD		
Wed, May 14	Enter intervention in https://goc.gridops.org and do Broadcast		
Wed, May 14	Take "snapshot" of C2ALICE Stager DB, rehearse upgrade		
Thu, May 8	Prepare CDB templates	Jan	

Detailed time table

Planned time	Responsible	Action	Comment
08:30	Ulrich	Pause batch jobs	
09:00	Jan	Stop CASTORALICE	
09:10	Nilo	Take DB snapshot	
09:30	Jan	Upgrade castoralice	including xrootd: https://twiki.cern.ch/twiki/bin/view/FIOgroup/CastorXrootdPlugin
	Nilo	Apply Oracle patches	?
	Jan	upgrade xrootd	
10:30		Test the instance	
11:00		Open service	
		Announce completion	

Notes

Tracking changes

- A difficult but common part of incident handling is to find out about all recent changes, even remotely related ones
  - Not necessarily changes to the affected service
- Process has to be lightweight and to a large part automated
  - Ideally a workflow with predefined and self-documenting state transitions
    - E.g. extract list of affected Configuration Items (nodes, devices, …)
    - May required deep level of site details
  - Access to change tracker ***must*** be authenticated and secure
- All changes are tracked, also standard (pre-authorized) changes
  - If something starts to go wrong at Site A on Day X
    - Anything changed at Site A on Day X?
    - Anything changed at Site B-Z on Day X?
    - Anything changed in the network on Day X?

WLCG Op meeting

- "Change Advisory Board" for changes impacting site availability?
  - Maintain and review list of scheduled changes with grid level impact
  - Each change is classified by the site in terms of
    - Impact: to site, to the grid, to a VO,…
    - Risk: likelihood of failure, ability to rollback, plan B, …
  - ***Authorize*** the change
    - Stakeholders agree that site can go ahead with the planning for the change
  - Periodically ***review*** list of executed changes
    - Define and assess according to simple KPIs (e.g. if or not the change caused an incident)
    - Goal is to improve the change mgmt process