CS Software
Consulting and Services GmbH

Two Easy (?) Pieces

Berichte aus dem Consulting Alltag

GTUG Stuttgart
27./28. September 2022

Two Easy (?) Pieces
Der Stand der Dinge

- „CS Software Consulting and Services“ ist nicht „CS Software Concepts and Solutions“
- … aber beide Unternehmen kooperieren eng
- Das „Consulting“ Unternehmen bietet Leistungen von mir
- … die auch gerne über die „Concepts“ bestellt werden können
- Die CS Software Consulting plant keine Produktangebote

Two Easy (?) Pieces
Eine Warnung

- Vortrag wird (zumindest in Teilen) sehr technisch
- Vorsicht, falls jemand gegen C allergisch ist
- Folien können Spuren von Quellcode enthalten

Two Easy (?) Pieces
Fall 1: Rien ne va plus

- Kunde betreibt komplexe Pathway Anwendung
- Ca. 150 Server (Cobol)
- Server können andere Server für fachliche Funktionen aufrufen
- Plötzlich Stillstand einzelner Anwendungsteile…
- …der sich im Verlauf der Zeit immer weiter ausbreitet
- Transaktionen laufen in AUTOABORT
- Schließlich Stillstand fast der kompletten Anwendung

Two Easy (?) Pieces
Fall 1: Rien ne va plus

- Lang laufende Transaktionen werden untersucht: o.B.
- Keine Deadlocks auf der Datenbank gefunden
- Starter lang laufender Transaktionen werden gestoppt – kein Erfolg
- Schließlich wird das gesamte Pathway System durchgestartet
- Danach läuft (vorerst…) alles wieder
- Das gleiche Problem tritt im Abstand von Wochen wiederholt auf

Two Easy (?) Pieces
Fall 1: Die Analyse beginnt

- Irgendwann werde ich angerufen, weil eine meiner Komponenten nicht mehr reagiert
- Ein erster Blick zeigt, dass der Prozess im SERVERCLASS_SEND_ hängt
- Analysetool: PSTATE – sollte jeder kennen!
- Der angesprochene Server hängt ebenfalls im SERVERCLASS_SEND_
- Alle SERVERCLASS_SEND_ Operationen waited und ohne Timeout
- In der Vergangenheit (… und die ist lang) gab es da nie Probleme
- Erst mit der Einführung der Anwendung, an der ich mitwirkte, starteten die Schwierigkeiten

Two Easy (?) Pieces
Fall 1: Die Analyse beginnt

- Wieder eskaliert das Problem, mehr und mehr Teile der Anwendung bleiben stehen
- Die Außenwirkungen werden sichtbar – und unangenehm
- NonStop wird eben „mission critical“ eingesetzt
- Analyse zeigt eine Kette von Servern, die einander aufrufen:
  A1 -> A2 -> A3 …
- Verfolgen der Kette sehr mühselig:
  - PSTATE zeigt, dass Prozess im SERVERCLASS_SEND_ steht
  - PSTATE zeigt das aufgerufene Cobol Unterprogramm
  - PSTATE zeigt nicht den gerufenen Server
  - Entwickler muss im Quellcode nachschauen
  - … während die Zeit läuft
- Irgendwann wurde entschieden, Pathway durchzustarten
- Danach lief es (vorerst) wieder

Two Easy (?) Pieces
Fall 1: Analyse - Vorgehen

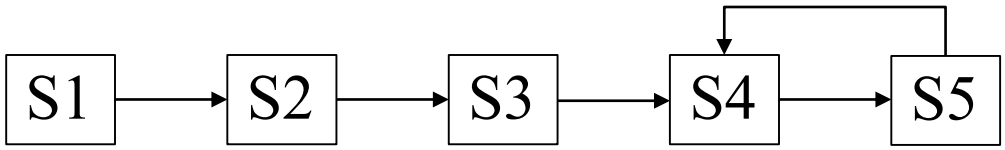
- Fehlerursache zunächst unbekannt
- Case bei HPE wurde geöffnet
  - Deadlock in Datenbank vermutet – widerlegt
  - Keine permanenten Locks zur Stillstandszeit
  - TMF Problem?
  - Pathsend Limit?
- HPE verlangte viele Infos beim nächsten Auftreten:
  - Measure Daten
  - Pathway Dump
  - CPU Dump
  - Eventlogs
  - usw.
- Kunde bereitete Skripte vor, um alle Informationen zuverlässig und vollständig bereitzustellen
- Arbeitsgruppe zur Problemlösung wurde gegründet – die Situation würde sicher wieder auftreten

Two Easy (?) Pieces
Fall 1: Analyse - Vorgehen

- Gleichartige Situation trat bereits nach wenigen Tagen erneut auf
- Alle von HPE geforderten Daten wurden gesammelt und an HPE geliefert
- Regelmäßige Calls mit HPE Support (auch mit Development)
- Die ersten Calls waren wenig ergiebig, Standardvorgehen bei HPE
  - Konzentration auf Measure Daten
  - Nicht wirklich hilfreich
- Arbeitsgruppe insistiert auf genauere Analyse der Pathsend Kette…
- …die HPE durch die CPU/Pathway Dumps auch machen kann
- Vermutlich ist das mühsam, da wir etwas Überzeugungsarbeit leisten mussten

Two Easy (?) Pieces
Fall 1: Gefahr erkannt

- HPE analysiert die Serverkette ausgehend von meinem „hängenden“ Prozess…
- … und findet dies:



- D.h. S1 löst eine Kette von SERVERCLASS_SEND_s aus…
- …die in einem Zyklus endet
- S4 wartet auf eine Antwort von S5 und S5 wartet auf Antwort von S4
- Klassischer Deadlock!

Two Easy (?) Pieces
Abschweifung: Fallen der Modularisierung

- Das ist ein Designproblem der Anwendung…
- Schon, aber ist es trivial, das zu entdecken?
- Ähnliche Probleme können in viel einfacheren Situationen auftreten…
- … und können eine Folge des (wünschenswerten) Code-Reuse sein

Two Easy (?) Pieces
Abschweifung: Fallen der Modularisierung

Beispiel:

Modul A kümmert sich um die Ausgabe von Fehlermeldungen

Modul B kümmert sich um die Konfiguration der Anwendung

Designer von B:
Wenn ich einen Fehler beim Lesen der Konfigurationsdaten feststelle, gebe ich eine Meldung aus. Dazu benutze ich eine Funktion von Modul B.

Designer von A:
Für maximale Flexibilität will ich konfigurieren können, welche Meldungen wo wie ausgegeben werden. Dazu benutze ich eine Funktion von Modul A.

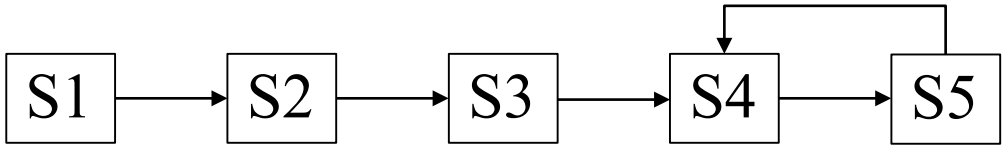
Also: B ruft A und A ruft B – potentiell ähnliche Situation!

In fachlichen Kontexten kann so etwas ebenso geschehen…
…und ist dann noch schwerer zu erkennen und noch viel schwerer zu beheben!

Two Easy (?) Pieces

Fall 1: Gefahr erkannt – noch nicht gebannt

- Bei erneutem Auftreten (die Situation wurden häufiger) wurden S4/S5 geprüft (PSTATE):



- Situation war identisch, bei gleichen S4/S5
- Stoppen der „hängenden“ Instanz von S5 löste den Deadlock sofort auf
- Also: Anweisung an Betrieb, was zu tun ist

Two Easy (?) Pieces

Fall 1: Was tun, sprach Zeus, die Götter sind besoffen

- Aber: Erkennung der Situation schwieriger als die Auflösung
- Stillstand von Anwendungsteilen wird spät bemerkt…
- … wenn schon Schaden entstanden ist
- Lösung des Designproblems kurzfristig schwierig:
  - Zu wenig Personal
  - Risiken neuer Fehler
- Andere, gleichartige Situationen sind möglich

Two Easy (?) Pieces
Fall 1: Die naheliegende Lösung

- SERVERCLASS_SEND_ mit Timeout als Übergangslösung
- Im Sourcecode: Änderungen an sehr vielen Stellen
- Neue Risiken
- Es gibt doch einen TIMEOUT Parameter in Pathway
- Kann so das Problem durch Konfiguration zumindest umgangen werden?
- Einfacher Test zeigt, dass TIMEOUT im Pathway zieht

Two Easy (?) Pieces
Fall 1: PATHWAY TIMEOUT – End of Story?

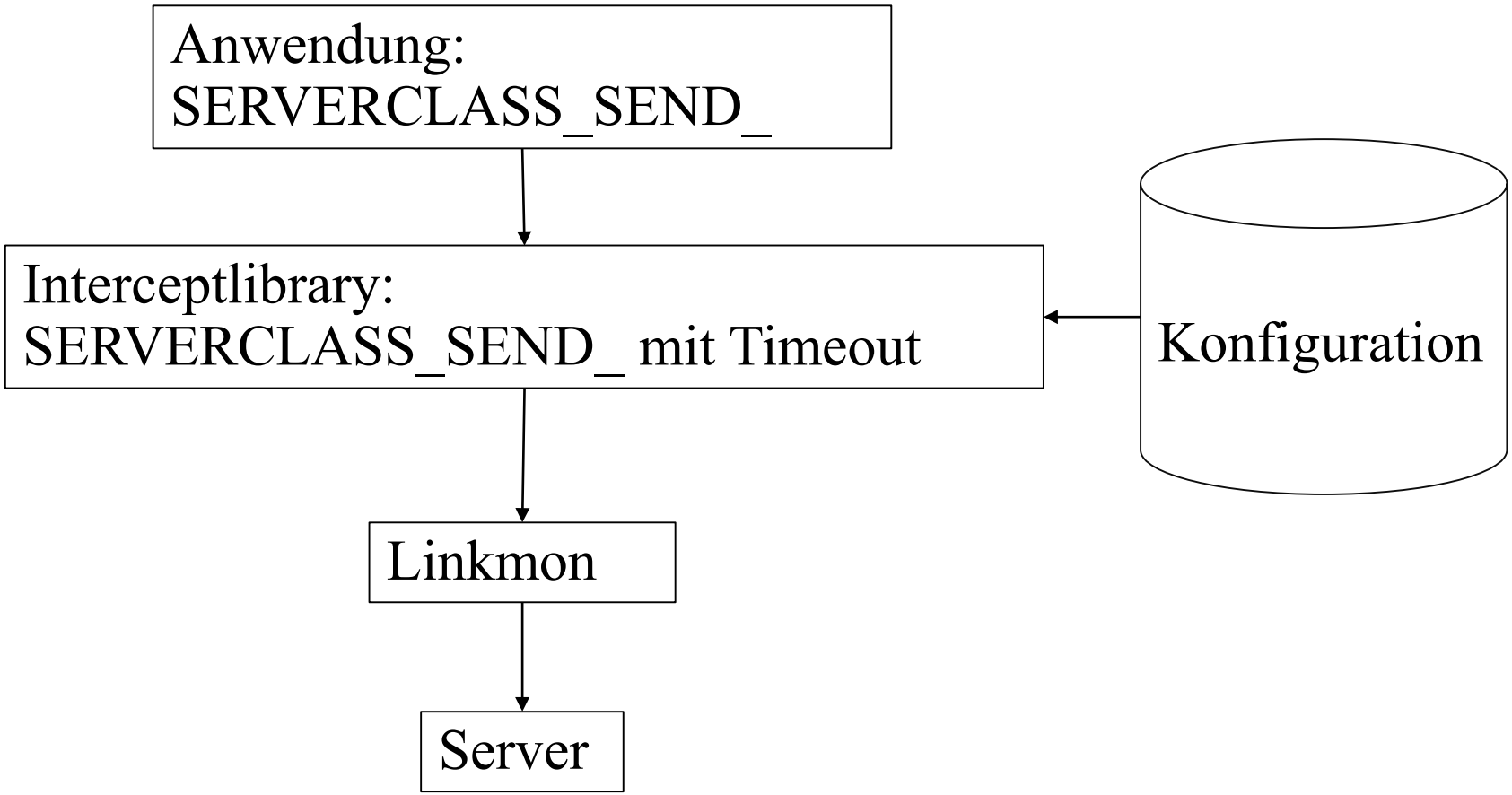
Ausführlichere Tests mit TIMEOUT Parameter:

- TIMEOUT funktioniert „meistens“
- TIMEOUT zieht bei bestehendem Link
- Neuer Link (neues Open des Linkmon auf Serverinstanz): Timeout von 5 Minuten
- Das wäre lang, aber noch akzeptabel
- Beobachtung: Fall, in dem trotz TIMEOUT SERVERCLASS_SEND_ „ewig“ hängt
- Fall von HPE Support nachvollzogen
- Hängt mit einem weiteren Open des Linkmons auf Pathmon zusammen
- Konsequenz: Pathway TIMEOUT nicht 100% zuverlässig und als Lösung ungeeignet

Two Easy (?) Pieces
Fall 1: Neuer Ansatz: Intercept Library

- Im Pathway kann beim Server eine Library konfiguriert werden
- Ist auch bei RUN Befehlen auf TACL Ebene möglich
- Die „USERLIBRARY“ kann auch Guardian Funktionen implementieren
- Laufzeitsystem verwendet die Implementierung der Userlibrary (falls vorhanden)
- Userlibrary kann Originalfunktion aufrufen…
- … u.U. mit anderen Parametern, z.B. Timeout
- Technik wird von etlichen Produkten verwendet:
  - Replikationstools
  - Securitytools (Greenhouse, Comforte)
- Userlibraries können kombiniert werden
  - Bereits vorhandene Userlibrary wird nur ergänzt
  - Funktioniert auch, wenn Userlibrary SERVERCLASS_SEND_ selbst benutzt

Two Easy (?) Pieces
Fall 1: Interceptlibray



Two Easy (?) Pieces

Fall 1: Konfiguration ist notwendig

- Einheitlicher Timeout für alle Server/Transaktionsarten nicht praktikabel
- Weitere Möglichkeiten für Zukunft gewünscht (Logging/Tracing)
- Konfiguration soll im Betrieb änderbar sein
- Änderung soll sofort wirksam werden
- Neues Lesen der Konfiguration bei jeder Aktion zu teuer

Two Easy (?) Pieces

Fall 1: Konfiguration ist notwendig

- Standardvorgehen:
  Bei Änderung der Konfiguration erhalten die Server eine Nachricht
- Hier nicht möglich, da keine Kontrolle über $RECEIVE in Userlibrary
- Guardian hilft: CONTROL 27
- Nowaited aufgerufenes CONTROL 27 komplettiert, wenn das zugehörige Enscribe File geändert wurde
- Also: Konfiguration in Enscribe (key sequenced) File, zum Setzen der Parameter Tool erforderlich

Two Easy (?) Pieces
Fall 1: Konfiguration ist notwendig

In der Intercept Library im SERVERCLASS_SEND_:

```
void CheckReadConfig(LibraryGlobals *sh)
{
  short fnum = sh->LG_ConfFile_NW;
…
  status = AWAITIOX(&fnum,,,,0 /* just check. */);
  if (_status_ne(status))
  {
 …
  } else
  {
    /* Control 27 hat komplettiert, also wurde ins Konfigfile geschrieben. */
   …
   read_config(sh);
  }
  status = CONTROL(sh->LG_ConfFile_NW,27,0);
  if (_status_ne(status))
  {
    …
  }
}
```

Two Easy (?) Pieces

Fall 1: Konfigurationstool: Auf den Schultern von Riesen…

- Simpler Kommandointerpreter
  - Anzeigen der Konfiguration
  - Setzen der Konfiguration
- Schreiben/Updaten genügt
  - alle Anwendungen bekommen das automatisch mit
- Kommandointerpreter aus vorhandenem Muster…
- …kann damit FC, Kommandohistorie, usw.

Two Easy (?) Pieces
Fall 1: Ein kleiner Adrenalinstoß

- Konzept implementiert, getestet
- Funktioniert …
- … aber (zunächst) nur für Cobol Server
- Problem: C Server arbeiten nowaited und rufen AWAITIO(-1,…)
- Completions auf dem Konfigfile
  - führen zu Fehlern (Filenummer unbekannt)
  - werden nicht in der Interceptlibrary gesehen
- Lösung: Auch Funktion AWAITIO muss „interceptet“ werden
- Im Intercept wird alles durchgereicht, außer auf Filenummer des Konfigfiles

Two Easy (?) Pieces
Fall 1: Ab in die Produktion

- Intercept Lösung wurde beschlossen
- Innerhalb von wenigen Tagen implementiert
- Mit dem Kunden getestet
- In die Produktion eingeführt
- Beschluss bis Einführung: ca. 1 Woche

Two Easy (?) Pieces
Fall 1: Ab in die Produktion

- Intercept Library macht $0 Ausgabe, wenn Timeout geschieht
- Tool des Kunden scannt $0 Logs und sendet E-Mail bei dieser Meldung
- Seit Einführung der Library kein Stillstand mehr
- E-Mails kommen immer mal wieder
- Gute Zwischenlösung bis Behebung an der Wurzel

CS Software Consulting and Services GmbH

Fragen?