Monitores Transaccionais

Tecnologias de Middleware

João Nogueira

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Setembro de 2006

Transacção

- Unidade de trabalho que executa apenas uma vez e produz resultados permanentes
- Pretende-se que cada transacção obedeça às propriedades ACID para garantir que é processada de forma fiável:
  - Atomicidade
  - Coerência (*consistency*)
  - Isolamento
  - Persistência (*durability*)

Transacção (2)

- Um pedido (de um terminal) pode necessitar a execução de uma ou mais transacções
- Cada transacção é delimitada pelas abstracções de primitivas:
  - Start-transaction
  - End-transaction
- End-transaction pode ter apenas dois resultados possíveis:
  - Commit (aceitar TODAS as operações efectuadas pela transacção)
  - Abort (descartar TODAS as operações efectuadas pela transacção)

Evolução dos Sistemas Distribuídos

- Acesso remoto
  - Terminais ligados a *mainframes* com sessões remotas
  - Primeiros monitores transaccionais
- Distribuição de ficheiros e memória
  - Aparecimento das *workstations*
- Acesso remoto II
  - Redução dos recursos das *workstations* e investimento em servidores de recursos (disco, processamento)
  - Aparecimento dos terminais X e *workstations* sem disco
- Arquitectura cliente-servidor
  - Serviços em servidores centrais, clientes com computações locais

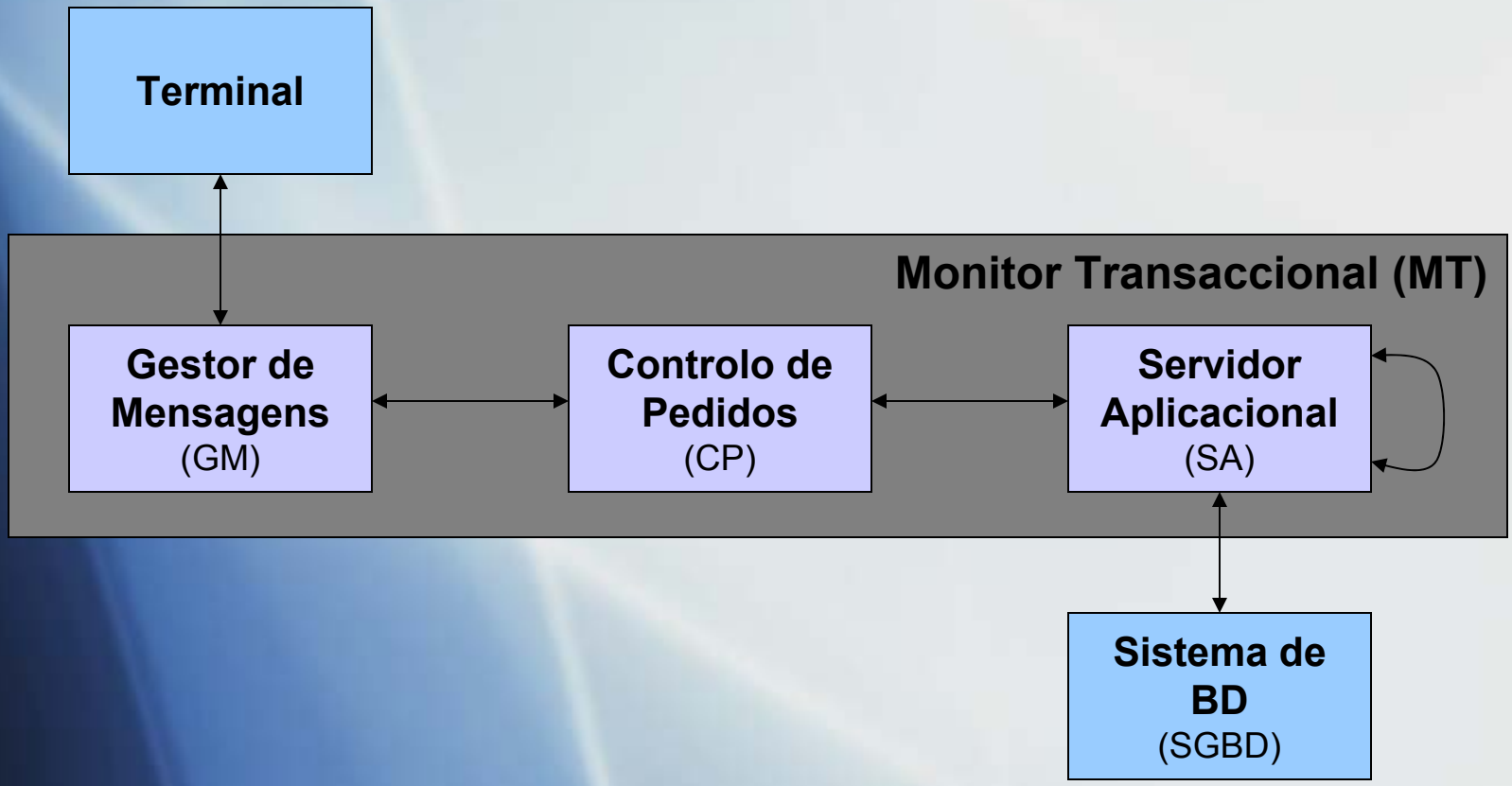
Evolução dos Sistemas Distribuídos (2)

- Arquitectura cliente-servidor 3-tier
  - Aparecimento dos *thin-clients*
  - Monitor Transaccional clássico apresentado insere-se neste modelo
- Arquitecturas de código móvel
  - e.g. Javascript, Java, ActiveX
- Arquitecturas de localização móvel
  - Permite a mobilidade de clientes e servidores
- Arquitecturas baseadas em eventos
  - e.g. bus de mensagens, produtor-consumidor

Monitor Transaccional Clássico (1990)

- Modelo de computação que suporta a abstracção de transacção
- Aproximação modular ao sistema que permite a separação de problemas por componentes do MT
- Exemplos de outros modelos de computação existentes na época que não suportam a noção de transacção:
  - Modelo de processamento em bloco (*batch*)
  - Modelo de divisão de tempo (*time-sharing*)
  - Modelo de tempo-real (*real-time*)

Monitores Transaccionais: Arquitectura Genérica



Monitores Transaccionais:
Diferenças para outros modelos (1)

- Em relação ao modelo de processamento em bloco (*batch*):
  - Suporte para a abstracção de transacção
  - Suporte para um grande número de terminais e utilizadores activos
  - Acesso aleatório e associativo a ficheiros
  - Tratamento de falhas de granularidade fina

Monitores Transaccionais:
Diferenças para outros modelos (2)

- Em relação ao modelo de divisão de tempo (*time-sharing*):
  - Suporte para a abstracção de transacção
  - Regularidade da carga das aplicações
  - Requer maior disponibilidade do sistema

MT Clássicos: Exemplos

- **ACMS** - Application Control and Management System
  - Digital
- **CICS** - Customer Information Control System
  - IBM
- **Pathway**
  - Tandem
- **IMS/DC** - Information Management System / Data Connection
  - IBM

Modelo MT
Fluxo de Execução Genérico

1) Interagir com o terminal para receber dados da transacção (geralmente através de formulários e menus **[GM]**
2) Traduzir os dados recebidos para uma mensagem de pedido num formato padrão **[GM]**
3) Iniciar a transacção **[CP]**
4) Examinar o cabeçalho do pedido de forma a determinar o seu tipo **[CP]**
5) Executar a aplicação referente ao tipo de pedido. Esta pode invocar outras aplicações e/ou operações sobre bases de dados **[SA]**
6) Finalizar a transacção (*commit* ou *abort*) depois de aplicação invocada terminar **[CP]**
7) Enviar o resultado da transacção para o terminal requerente **[GM]**

Modelo MT

Comunicação entre componentes

- Um sistema tem, tipicamente, várias instâncias de GM's, CP's e SA's
- Estas instâncias seguem o paradigma de comunicação imposto pelo MT:
  - GM's comunicam com CP's
  - CP's comunicam com SA's
  - SA's comunicam com SGBD's e outras SA's

Modelo MT
Mapeamento de Componentes no SO

- A arquitectura modular do MT facilita a programação das aplicações:
  - Componentes são mapeados em processos do SO
  - Oferece meios de comunicação entre eles
  - Oferece funcionalidades de monitorização e gestão do desempenho, faltas e segurança

Gestor de Mensagens
(Message Manager)

- Componente que faz a interface com os terminais
- Funções principais:
  - Formatar pedidos
  - Gerir formulários
  - Validar input
  - Gerar output
  - Fazer verificações de segurança

GM - Formatação de Pedidos

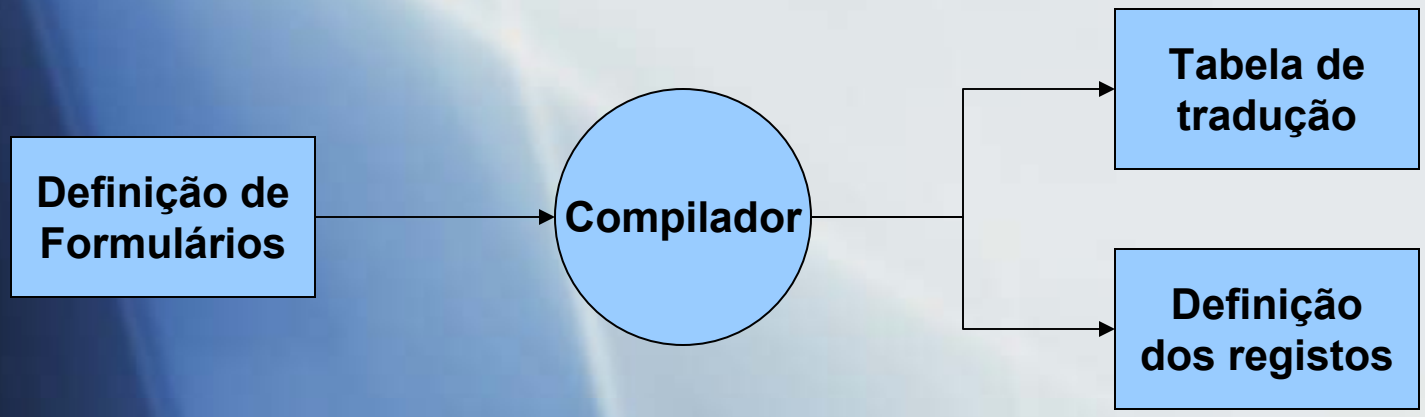
- O formato dos pedidos a tratar pelos CP's é definido pelo MT e inclui:
  - Um cabeçalho padrão, específico de cada MT, mas igual para todas as aplicações de cada sistema
  - Um corpo definido por cada aplicação

GM - Gestão de Formulários

- Para isolar os CP da diversidade de interfaces oferecidos pelos diferentes terminais, o GM traduz os inputs daqueles para o formato padrão dos pedidos que os CP entendem
- Desta forma, consegue-se independência do tipo de terminal por parte do MT

GM - Gestor de Formulários

- Componente do GM que inclui:
  - Um editor para que o programador das aplicações possa definir e modificar formulários
  - Um compilador cria a definição dos registos e uma tabela de tradução a partir da definição dos formulários criada pelo programador



GM - Gestor de Formulários (2)

- A tabela de tradução é uma versão compilada do formulário, usado pelo Gestor de Formulários em tempo de execução para traduzir do formato específico de cada terminal para o formado padrão
- A definição dos registos é a descrição do formulário numa linguagem de alto nível que pode ser usada por uma aplicação para o interpretar

GM - Validação de Input

- O GM é também responsável pela validação dos dados passados pelo terminal.
- Pode verificar:
  - Se o valor de cada campo é do tipo correcto
  - Se o valor de cada campo está semanticamente correcto
- Durante a verificação não acede a bases de dados actualizadas pelas transacções, mas pode usar uma imagem (*snapshot*) daquelas

GM - Produção de Output

- Muitas vezes a execução de um pedido produz resultados a devolver ao terminal requerente:
  - Este pode ser apresentado ao utilizador do terminal, ou
  - Ser um comando especial a enviar a um dispositivo (e.g. abrir a gaveta do dinheiro)
- O GM encarrega-se de enviar a devida informação de volta ao terminal / dispositivo

GM - Segurança

- Tipicamente são feitas também algumas verificações de segurança:
  - Autenticação de cada utilizador (e.g. *password*)
  - Controlo de acesso, apenas deixando fazer pedidos a que o utilizador está autorizado

- Esta função deve ser executada pelo MT pois o SO e o SGBD não têm informação acerca dos privilégios de cada utilizador

Controlo de Pedidos
(Request Control)

- Cada pedido construído pelo GM é passado a um CP para ser atendido
- O CP é responsável pelo encaminhamento destes para o SA correcto que o irá processar
  - Esse SA é identificado através da leitura do cabeçalho do pedido, onde está especificado o seu tipo

Servidores Aplicacionais (Application Servers)

- Um SA consiste num ou mais programas que podem, tipicamente, aceder a bases de dados ou a outros SAs
- Têm como função fazer o processamento dos pedidos em si
  - Por exemplo alterar os registos bancários de dois clientes para efectuar uma transferência de fundos

Mapeamento CP/SA

- O mapeamento, dentro do CP, entre tipos de pedidos e SA's deve ser dinâmico
  - Facilita a tolerância a faltas
  - Facilita a (re)distribuição de componentes
  - Torna o sistema mais escalável
- Uma tabela local ao CP é uma forma simples e eficaz de implementar isto, mas...
  - Se existe mais que uma cópia do CP, e se CP's diferentes atendem pedidos de tipos diferentes, um CP pode receber um pedido que não sabe como atender

Mapeamento CP/SA:
Formas de mapeamento (1)

- Cada GM sabe que tipos de pedidos cada CP tem capacidade de tratar
  - O GM está programado para enviar cada pedido para um CP que o consegue atender

Mapeamento CP/SA:
Formas de mapeamento (2)

- Cada CP conhece os tipos de pedidos que cada outro CP tem a capacidade de atender
  - O GM envia cada pedido para um CP qualquer e este encarrega-se de o reencaminhar, se necessário

Mapeamento CP/SA:
Formas de mapeamento (3)

- O sistema suporta um **serviço de nomes**:
  - É acessível a qualquer nó
  - Pode ser usado para mapear tipos de serviço a CP's que os podem atender
  - Tanto os GM como os CP podem tomar a responsabilidade de consultar o serviço de nomes e (re)encaminhar cada pedido para o CP correcto

Comunicação CP/SA

- Para chamar um SA, um CP deve algo que o identifique
- A natureza desse identificador é determinada pelo SO e pela arquitectura de comunicação usada pelo MT
- Os mecanismos de **gestão de processos** do MT definem essas ligações entre componentes

Comunicação CP/SA:
Distinção entre CP e SA

- Alguns MT não distinguem CP's de SA's:
  - Usando um serviço de nomes, é possível mapear tipos de pedidos a SA's directamente
  - Não existem CP's, só SA's
  - Alguns SA's encaminham pedidos para outros (fazem a função de CP's). Estes últimos apenas processam os pedidos

Comunicação CP/SA:
Fluxo e desempenho

- Esta estrutura pode minimizar o número de associações entre componentes:
  - Cada GM só comunica com um (ou poucos) SA's que fazem a função de CP
  - Cada SA que processa pedidos só comunica com aqueles que fazem a função de CP
  - Para minimizar re-associações, estas são, tipicamente, de longa duração

Gestão de Processos

- Uma das funções do MT é definir os mecanismos de gestão para a criação e manutenção dos processos para GM's, CP's e SA's
  - Um processo é uma abstracção do SO que consiste num espaço de endereçamento, num estado do processador e num conjunto de recursos (e.g. processo Unix)
- Há várias estratégias de gestão de processos

GP - MT Centralizado

- Neste modelo, todos os componentes estão a executar na mesma máquina (e.g. *mainframe*) e encontram-se no mesmo espaço de endereçamento
- Há duas aproximações a este modelo:
  - *Single-threading*
  - *Multi-threading*

GP - MT Centralizado:
Single-threading

- Um processo por terminal
  - Executa uma *imagem* do MT que consiste nos GM, CP e SA necessários
  - Comunicação é feita usando os mecanismos de IPC oferecidos pelo SO
  - Pouco escalável
  - Difícil de controlar a distribuição de carga

GP - MT Centralizado:
Multi-Threading

- Um único processo gere todos os terminais
- Conceptualmente cada terminal está associado a uma *thread* de controlo própria:
  - Todas partilham o mesmo espaço de endereçamento
  - Tipicamente implementadas pelo SO ou pelo MT
  - O sistema alterna entre as *threads* para atender diferentes terminais (mais eficiente que processos)
  - Sistema de *threads* deve interceptar as chamadas bloqueantes ao sistema e implementar trocas de contexto entre *threads*

GP - MT Distribuído

- MT pode tomar a forma de um sistema distribuído, por várias razões:
  - Eficiência
  - Escalabilidade
  - Dispersão geográfica
  - Questões de gestão e administrativas

GP - MT Distribuído (2)

- Como há, tipicamente, mais comunicação entre um terminal e o seu GM que entre o GM e o CP, é eficiente passar os GM's para perto dos terminais
  - Caso os terminais tenham algum poder de processamento (e.g. workstations)
- Como os CP e SA podem estar dispersos pela rede, muitas das operações entre componentes serão agora remotas e requerem algum tipo de comunicação

GP - MT Distribuído:
Comunicação Entre Processos

- A comunicação entre processos (remotos e não remotos) pode tomar várias formas que correspondem a diferentes níveis de abstracção:
  - Troca de mensagens
  - Chamadas remotas (*Remote Procedure Calls - RPCs*)
  - Arquitectura cliente-servidor recursiva

GP - MT Distribuído:
Troca de Mensagens

- É estabelecida uma ligação (e.g. sessão, circuito virtual). Depois disso os processos podem trocar mensagens.
  - Usado no IBM CICS
  - Cada processo pode enviar ou receber mensagens conforme necessário
  - A ligação deve ser de longa duração, sendo usada por várias transacções
  - Tem a vantagem de não impor uma estrutura rígida ao formato da comunicação
  - Tem a desvantagem de as aplicações terem que implementar todo o protocolo de comunicação (sintaxe e semântica).
  - A associação entre componentes é tipicamente definida estaticamente, logo qualquer alteração (e.g. endereços) implica uma alteração das aplicações

GP - MT Distribuído:
RPC's

- A disparidade entre a comunicação entre processos com troca de mensagens e intra-processo (invocações de funções) pode ser escondida usando este paradigma:
  - Facilita a tarefa do programador pois todas as chamadas (locais e remotas) passam a ter o mesmo formato
  - Evita a modificação do programa a cada alteração da máquina onde corre um determinado componente
  - Usa o paradigma pedido-resposta, evitando alguns erros de programação mas diminuindo a flexibilidade na estrutura da troca de mensagens: esta é definida e suportada pelo MT que implementa o RPC
  - Esconde grande parte dos problemas e peculiaridades de comunicação entre processos

GP - MT Distribuído:
Modelo Cliente-Servidor Recursivo

- Alguns processos podem, no paradigma RPC, ser simultaneamente clientes e servidores:
  - e.g. CP pode ser servidor para GM e cliente para SA
  - Os RPC's podem ser usados de forma recursiva para adaptar o sistema a este modelo
  - Usado pelos Tandem Pathway e Digital ACMS
- Isto indroduz uma facilidade adicional de reconfiguração e programação, mesmo em componentes que residem no mesmo espaço de endereçamento
  - O paradigma de comunicação entre todos os processos (locais e remotos) é, agora, sempre o mesmo: invocação de funções através de troca de mensagens. A localização é transparente para o programador

GP - MT Distribuído:
Modelo Cliente-Servidor Recursivo (2)

- O custo, em termos de número de instruções deste novo modelo é muito superior às invocações de funções locais clássicas:
  - 1000-10000 vs ~50 instruções
- Outro problema de desempenho é o uso de mais processos que no modelo tradicional:
  - Mais trocas de contexto
  - Este problema pode ser minimizado usando, mais uma vez, *multi-threading*

GP - MT Distribuído:
Multi-Threading

- Semelhante ao modelo *multi-threaded* no MT centralizado
- Problema das chamadas bloqueantes, se as *threads* forem implementadas pelo MT pode ser resolvido introduzindo restrições ao modelo:
  - *Multi-threading* para GM's e CP's, mas não podem invocar operações sobre SGBD
  - MT intercepta pedidos a outros processos, tornando-os assíncronos
  - SA têm que ser *single-threaded* mas podem invocar serviços de forma síncrona
  - SA's podem-se tornar ponto de estrangulamento: podem ser replicados para melhorar o desempenho mas os CP têm que saber quais estão ocupados e quais estão livres...

GP - MT Distribuído:
Classes de SA's

- Para resolver o problema anterior, alguns MT implementam a abstracção de classes de SA's:
  - Conjunto de processos que executa o mesmo programa SA
  - Funciona como coordenador dos processos que gere, redireccionando os pedidos para os SA que os podem atender mais rapidamente
  - Tem uma fila de pedidos partilhada por todos os SA de uma classe

GP - MT Distribuído:
Filas

- Outro problema de comunicação deve-se ao facto de os componentes poderem falhar de forma independente
  - A falha de um servidor não deve impedir o funcionamento de um cliente
  - Os MT podem lidar com este problema usando filas nas comunicações
  - Mesmo sem falhas, pode ser impossível executar uma determinada transacção imediatamente quando é requerida devido a ocupação dos recursos necessários

GP - MT Distribuído:
Filas (2)

- Filas podem ser usadas para garantir fiabilidade:
  - Guardar todos os pedidos, por ordem, em armazenamento estável antes de devolver as respectivas confirmações
  - As transacções que executam os pedidos retiram-nos da fila. Se aquelas abortam, esta operação é revertida
- Estes sistemas normalmente incorporam mecanismos de escalonamento
  - Podem ser atribuídas prioridades a tipos de pedidos ou pedidos específicos
  - Escalonador colocado antes dos CP's pode atribuir pedidos a CP's específicos com base num critério qualquer

GP - MT Distribuído:
Filas (3)

- O tamanho das filas é uma medida dos recursos necessários a este sistema
- A maior desvantagem é o desempenho:
  - O processo de colocar um pedido na fila para mais tarde ser retirado é mais dispendioso que o envio deste directamente do GM para o CP
- Este tipo de serviço é oferecido pela maior parte dos MT
  - Por vezes como opcional (e.g. IBM CICS ou Digital ACMS)
  - Outras como a base do mecanismo de comunicação (e.g. IBM IMS/DC e Digital DECintact)

Gestão do Sistema

- Para manter o sistema a funcionar correctamente, este deve oferecer ferramentas de gestão:
  - Monitorização e ajustamento de desempenho, faltas, segurança, configuração, etc.
- Especialmente importante em sistemas distribuídos
- O mais automatizado possível

Gestão do Sistema (2)

- A abstracção de transacção e as filas de pedidos possibilitam que a recuperação de componentes seja transparente:
  - Quando um processo falha, as transacções que estavam a executar neste abortam
  - Quando o MT recupera o processo falhado, os pedidos que correspondem às transacções abortadas recomeçam automaticamente

Evolução dos MT Clássicos

- Naquela altura (~1990), os MT vieram resolver um conjunto de problemas dos sistemas distribuídos, de forma metódica, que não eram tratados pelos sistemas sobre os quais este *middleware* se situava: SO's, SGBD e redes
- Começou a verificar-se uma tendência para incluir algumas dessas funcionalidades apenas oferecidas pelos MT nos próprios sistemas base:
  - *Multi-threading* e RPC's nos SO's
  - Transacções mais completas nos SGBD (e.g. *2-phase commit*)
- Isto levou a uma simplificação dos MT, possibilitando que os seus criadores se focassem mais nos desenvolvimento de novas funcionalidades e modelos mais evoluídos

CICS

Customer Information Control System

- Monitor Transaccional da IBM
- Primeira versão: 8 Julho 1969
- Usado em aplicações bancárias, ATM's, reservas de bilhetes de avião, etc.
- Melhoramentos recentes incluem suporte para *WebServices* e *Enterprise Java Beans*
- Versão mais recente (3.1) data do início de 2005

CICS
Componentes do MT

- Clientes CICS:
  - Aplicação principal usada para associar o nó a uma região CICS
  - Transacções executam através do cliente CICS
- Uma região CICS é um conjunto de recursos controlado pelo CICS como um todo

CICS
Componentes do MT (2)

- Escalonador de transacções:
  - Responsável por escalonar e enviar para os SA's as transacções que aguardam execução, controlar a carga de trabalho do sistema e o número de SA's.
  - Cria ou destrói SA's, de acordo com a carga do sistema, dentro de limites máximo e mínimo pré-definidos
  - Cada utilizador, dispositivo e transacção tem uma prioridade associada
  - O escalonador calcula uma prioridade geral de cada pedido usada em caso de carga do sistema

CICS
Componentes do MT (3)

- Servidores Aplicacionais:
  - Um SA é um processo *multi-threaded* do CICS que oferece um ambiente completo para executar transacções
  - Uma transacção pode ser executada por um só SA ou por vários, cooperando de forma sequencial
  - Todas as SA's devem estar no mesmo nó, mas este pode não ser o mesmo do cliente CICS
    - Comunicam usando sinais e memória partilhada

CICS
Componentes do MT (4)

- Structured File Server (SFS)
  - Sistema de ficheiros para uso das aplicações
  - Pode existir mais que um e/ou, opcionalmente, uma base de dados relacional
  - Oferece funcionalidades padrão de acesso aos ficheiros e diversos tipos de filas de dados
  - O acesso a SFS's é sempre feito pelos SA's:
    - Comunicação é feita através de RPC's

CICS
Componentes do MT (5)

- PPC Gateway Server
  - Interface de acesso a redes SNA
  - Comunica com o CICS usando TCP/IP
  - Aplicações que o usam são clientes do PPC Gateway Server

CICS
Lógica de Programação

- O desenho de aplicações implica a sua divisão em módulos com lógica distinta:
  - Serviços de apresentação
  - Serviços de dados
  - Lógica de negócio
  - Lógica de determinação de problemas

CICS
Exemplos de Comandos Disponíveis

- Chamar outro programa:
  - LINK, RETURN
- Tempo:
  - ASKTIME, CANCEL, FORMATTIME
- Sincronização (acesso serializado a recursos):
  - ENQ, DEQ
- Delimitação de unidades lógicas de uma transacção:
  - RETURN, SYNCPOINT
- Comunicação:
  - SEND, RECEIVE, ALLOCATE, FREE
- Monitorização do desempenho:
  - COLLECT STATISTICS, INQUIRE STATISTICS

CICS
Criação de um *HelloWorld*

- Código C (WORLDPRG.ccs):

```
int main()
{
    char Hello[] = "HELLO WORLD!";
    EXEC CICS SEND FROM(Hello) LENGTH(12) ERASE;
    EXEC CICS RETURN;
}
```

- Compilar:
  - ciccstcl -lC WORLDPRG.ccs
  - Gera um executável WORLDPRG (ou WORLDPRG.dll em Windows)

CICS
Execução do *HelloWorld*

- O administrador CICS introduz o programa no sistema:
  - Adiciona uma referência para o programa WORLDPRG na região CICS
  - Adiciona uma referência para a transacção HELO para executar esse programa nessa região

- Executar a transacção HELO:
  - Executar **`cisctermm`** para abrir um cliente e ligar ao sistema
  - Lançar o comando **`HELO`**
  - Deve aparecer o texto "`HELLO WORLD!`" no terminal

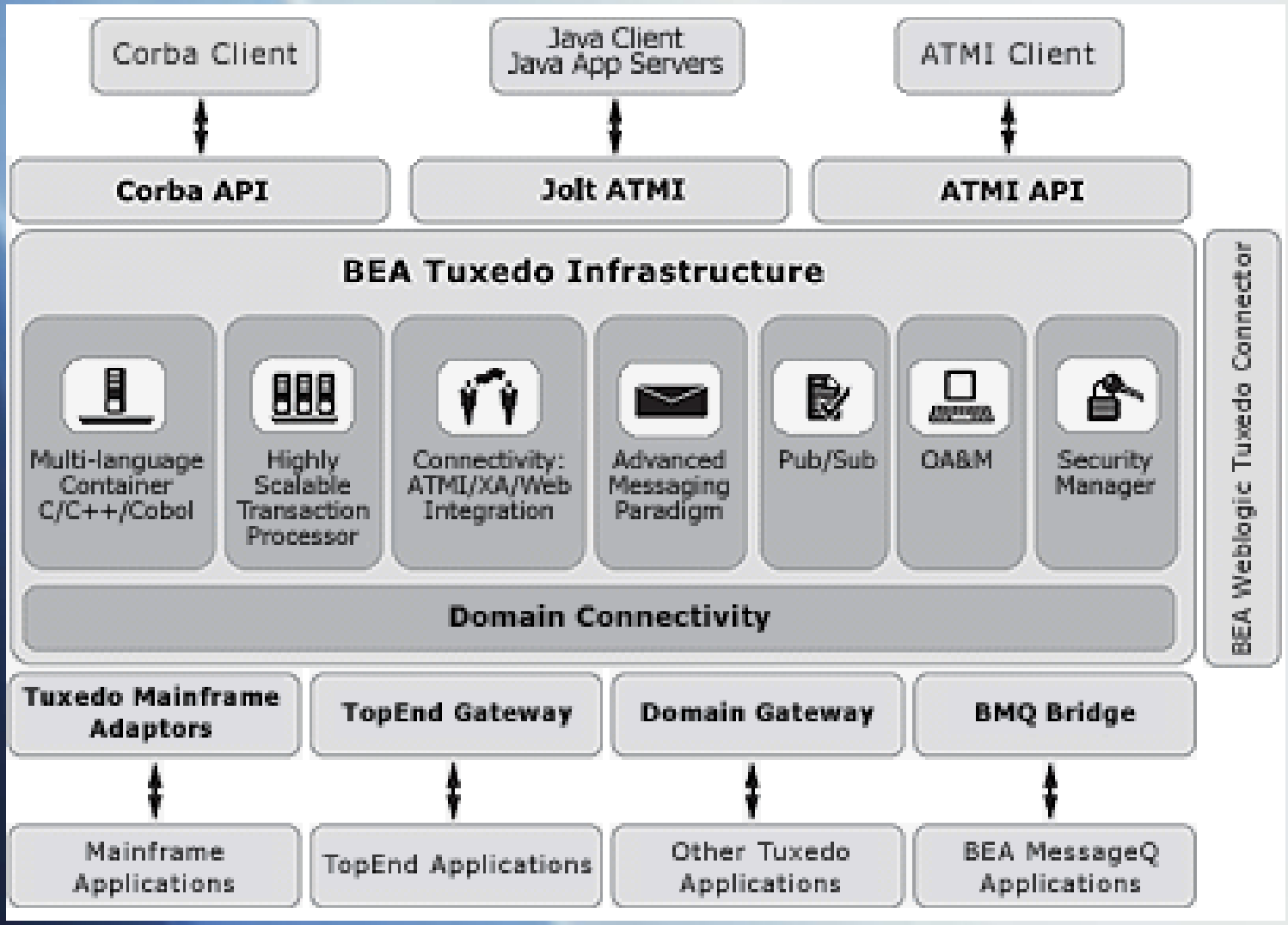
MT Modernos

- Adaptaram-se às novas arquitecturas de sistemas distribuídos:
  - Poder de processamento nos terminais (PCs, *workstations*), tipicamente executam a aplicação cliente - *browser*
  - Terminais capazes de interpretar formas complexas de apresentação de dados e formulários (HTML, CSS, XML, scripts php, .Net, etc.) e incluir nesta porções de código migrado do servidor (Javascript, Java, ActiveX, Flash, etc.)

MT Modernos (2)

- Tipicamente 3- (ou mais) tier, separando:
  - Geração de dados/código de apresentação
  - Lógica de negócio
  - Serviços de dados (ficheiros, bases de dados, etc)
- Invocação de serviços pode ser recursiva:
  - e.g. serviço de reservas contacta do aeroporto contacta companhias aéreas, rent-a-car's, hotéis, etc. para satisfazer o pedido de um cliente
  - Tudo isto é transparente para o cliente
- Comunicação pode usar diversos paradigmas:
  - Troca de mensagens, proxies, object-brokers, RPC's, sistemas produtor-consumidor, etc.
- Maior foco na segurança

MT Modernos: Bea Tuxedo



Referências

- P. Bernstein, **Transaction Processing Monitors**, *Communications of the ACM, Vol. 33, N. 11*. Nov. 1990
- IBM, **CICS Application Programming Guide**, version 5.1, 1999
- P. Veríssimo, L. Rodrigues, **Distributed Systems for System Architects**, Kluwer Academic Publishers, 2001

- **IBM CICS Website**, http://www.ibm.com/cics, Set. 2006
- **Bea Tuxedo Website**, http://www.bea.com/tuxedo, Set. 2006