5
Uma Extensão para Reserva de Recursos no CSBase

A habilidade de infra-estruturas de *software* para aglomerados de computadores em oferecer garantias de tempo no processamento das aplicações é um dos principais fatores de aceitação para as diversas aplicações distribuídas. As restrições nas garantias de tempo de processamento são:

- Ser obtida em plataformas de propósito geral;
- Ser controlável no mesmo nível da aplicação (*application-level*).

Motivada por estas exigências de aceitação dos usuários, a pesquisa em sistemas operacionais distribuídos sempre buscou maneiras de combinar uma coleção de recursos computacionais distribuídos de uma forma homogênea e transparente para os usuários. Diversos sistemas operacionais originalmente concebidos para serem distribuídos [35], permitem que os usuários executem as suas aplicações distribuídas, da mesma forma em que executariam em uma máquina local isolada. Nesses sistemas operacionais a distribuição dos diversos processos dos usuários para as várias máquinas é feita transparentemente pelo sistema, sem que o usuário tenha que se preocupar com isso. Esta abordagem traz inúmeras vantagens e permite que os recursos computacionais disponíveis sejam melhor aproveitados.

Com a mesma motivação desses sistemas operacionais, para o *framework* CSBase (que usa sistemas operacionais tradicionais como UNIX e Windows) uma questão chave, assim como para outros sistemas para aglomerados de computadores, é a utilização eficiente da capacidade de processamento disponível nas máquinas sob seu gerenciamento.

Originalmente, o CSBase se limitava a monitorar a capacidade ociosa de processamento nessas máquinas e a utilizar essa informação para a seleção de máquinas na execução de novas aplicações. Entretanto, as primeiras aplicações desenvolvidas com o CSBase já demonstraram que são necessários mecanismos que permitam um melhor gerenciamento e controle dos recursos computacionais disponíveis em ambientes distribuídos, e especialmente em

aglomerados de computadores dedicados à execução de aplicações de alto desempenho.

Neste trabalho, apresenta-se uma extensão ao *framework* CSBase que possibilita a reserva de processador para aplicações de usuários do sistema, permitindo assim um gerenciamento mais eficiente dos recursos computacionais disponíveis nos SGA's. Essa extensão também garante que serão efetuadas as adaptações necessárias para acomodar eventuais variações no perfil de uso do processador por parte das aplicações.

A escolha de onde os componentes são executadas é feita de forma implícita ou explícita pelo usuário, e leva em conta características estáticas e dinâmicas do *hardware* e do *software* disponível no sistema distribuído. Além disso, usuários e programadores podem (e devem !) também especificar os requisitos de utilização de recursos dos componentes de *software*. A partir dessa especificação, o sistema se encarrega de alocar uma máquina apropriada para executar o componente e de garantir o tempo de processamento através da reserva dos recursos necessários.

À seguir descreve-se a extensão do *framework* CSBase, com a incorporação da funcionalidade do novo serviço baseado em objetos distribuídos, que dá suporte para o gerenciamento e a reserva de processador em sistemas distribuídos. Essas novas funcionalidades oferecem benefícios para: (i) aglomerados (*clusters*); (ii) sistemas de computação em grade (*grid computers*); (iii) sistemas operacionais distribuídos; e (iv) sistemas de *middleware* baseados em objetos distribuídos com suporte para qualidade de serviço (QoS). Além disso, como a funcionalidade é baseada em CORBA, ela pode interagir com outros serviços e aplicações CORBA e ser executada em cima de diversos sistemas operacionais e plataformas de *hardware*.

Uma observação importante é que, por questões operacionais, os experimentos foram efetuados utilizando a plataforma Windows 2000 5.00.2195, *service pack* 4. Contudo, como a API de integração é única para as demais plataforma e, as interfaces são aderentes ao modelo CORBA, conferindo portabilidade e escalabilidade ao ambiente, teoricamente o novo serviço pode ser utilizado em qualquer outra plataforma.

5.1. Integração

Na composição da extensão do *framework* CSBase, foi utilizado o sistema DSRT para verificação, admissão e reserva de recursos, assim como nas adaptações necessárias em tempo de execução da aplicação. Sua utilização neste trabalho deve-se ao fato deste sistema oferecer uma interface padrão para a obtenção de informações dinâmicas sobre o estado dos recursos em uma máquina. Provê ainda suporte para controle de admissão de novos processos baseado em seus requisitos não funcionais, reserva de recursos (processador) e escalonamento de tempo real flexível (*soft real-time*).

A integração do *framework* CSBase com o sistema DSRT possibilitou uma melhor utilização dos recursos computacionais disponíveis. Passou a ser ofertado à aplicação cliente, um mecanismo capaz de reservar os recursos necessários a ela assim como, adaptá-la dinamicamente a seu ambiente de execução.

Com essa integração, buscou-se disponibilizar um novo serviço de reserva e gerência de processador, projetado para suportar adaptações nas aplicações, assim como uma gerência eficiente de recursos em sistemas distribuídos dinâmicos e heterogêneos. Trata-se da integração entre a monitoração de recursos distribuídos, a iniciação remota de objetos CORBA, e um mecanismo de reserva de recursos.

5.2. Implementação da integração

Neste item será descrito como foi implementada a integração dos diversos sistemas e ferramentas envolvidas.

5.2.1. Repositório de interfaces

O *framework* CSBase utiliza o sistema LuaORB [2] [6] (Figura 18) para obter dinamicamente informações sobre os objetos e a descrição de suas interfaces, através de chamadas ao Repositório de Interfaces, que mantêm uma descrição acessível das interfaces dos servidores disponíveis. LuaOrb é baseada na interface dinâmica de chamada (DII - *Dynamic Invocation Interface*) de CORBA, mapeando seu caráter dinâmico no sistema de tipos dinâmicos de Lua [20], evitando assim o uso de *stubs*.

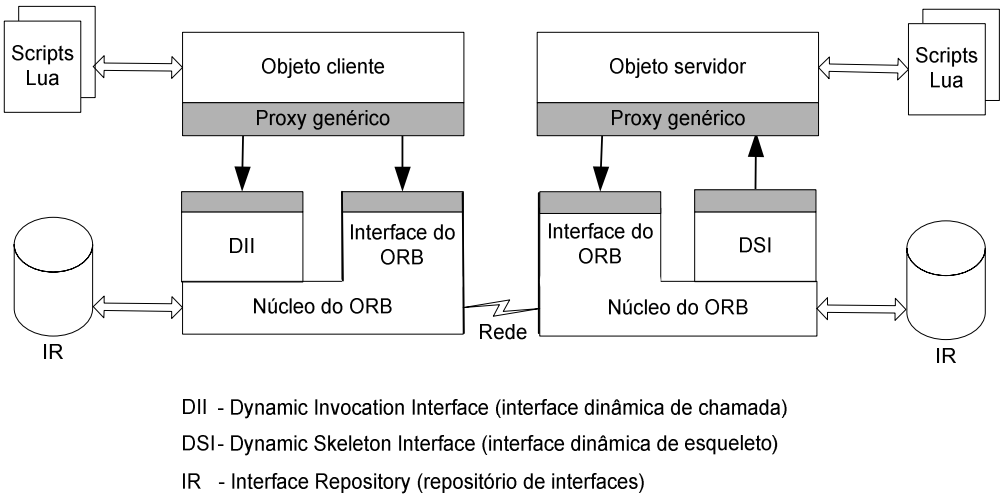


Figura 18 - LuaORB

5.2.2. IDL CSBase estendia

No desenvolvimento da extensão do *framework* CSBase, possibilitando a reserva de processador no SGA, foi modificada a interface do *daemon* SGA (SGA *Server* – arquivo sga-daemon.idl). No Anexo 1, tem-se esta interface de gerência do *framework* CSBase estendido, com a inclusão dos parâmetros lPeriod e dPercentage na operação executeCommand. Esta operação efetua a requisição de execução de uma aplicação. A inclusão desses novos operandos possibilita que ela ofereça a reserva de processador às aplicações.

O *framework* CSBase também utiliza o sistema LuaORB para efetuar chamadas CORBA. Na utilização do sistema LuaORB, quando o programador necessita chamar um método em um objeto CORBA, escreve apenas a chamada, usando a sintaxe regular para chamada de método de Lua. LuaOrb intercepta a chamada, mapeia dinamicamente os parâmetros de tipos de Lua para a IDL, faz a chamada real, e re-mapeia os resultados (no retorno) para Lua. Com LuaOrb, o usuário tem o acesso transparente a todos os objetos CORBA disponíveis no servidor, como se fossem objetos locais. Não há nenhuma necessidade de predefinir cabeçalhos (*headers*) de IDL e *stubs* para cada objeto. A publicação da IDL é realizada na mesma forma definida pelo modelo CORBA (Figura 19).

```
idl -I${LUAORB2_MICO}/include --feed-ir --feed-included-defs -
ORBIfaceRepoAddr inet: TARSILA:8888 sga-daemon.idl
```

Figura 19 - Publicação da IDL de gerência do *framework* CSBase (arquivo sga-daemon.idl)

5.2.3. Serviço no sistema operacional

Apresenta-se a seguir, o conceito básico e as principais funcionalidades oferecidas pelos serviços no sistema operacional. Não se pretende aqui fornecer uma descrição completa desse tema, que pode ser encontrada nas documentações específicas de cada sistema operacional, mas sim introduzir a terminologia e funcionalidade necessária à compreensão.

No *framework* CSBase, o Servidor de Gerência de Algoritmos (SGA) é um serviço instalado em cada máquina hospedeira. Serviços no sistema operacional são elementos que possibilitam a criação de aplicação executáveis de longa duração, com sessão própria. Sua principal finalidade é executar processos em modo não interativo (*background*). Esses serviços (processos) podem ser instanciados automaticamente quando a máquina é inicializada (*boot*), podem ser interrompidos temporariamente (*pause*) e posteriormente reiniciados (*restart*), podem ainda, ser instanciados manualmente e, tipicamente não interagem com o usuário através de alguma interface. A razão principal para isso é que os serviços no sistema operacional podem (e devem !) executar mesmo que não exista nenhum usuário trabalhando (*logged*) na máquina. Isso os torna ideais para o uso em servidores ou quando se necessita de alguma funcionalidade de longa duração, que não interfira nos processos de outros usuários que estejam trabalhando na mesma máquina.

A Figura 20 ilustra a interface do serviço de gerência do *framework* CSBase. O Anexo 2 mostra a implementação desta interface.

```
... ... ...
function SGA_DAEMON:executeCommand( comm_string, comm_id, host, Period, Percentage)
  if host == "" then host = nil end
  local comm = Command{
     command = comm_string,
     id = comm_id,
     host = host,
     Period = Period,
     Percentage = Percentage,
  }
  if comm then COMMAND:addCommandToList(comm) end
  return comm
end
... ... ...
```

Figura 20 - Interface do serviço (arquivo sga-services.lua)

5.2.4. Exportação de funções C → Lua

Funções definidas em Lua – e funções C registradas em Lua – podem ser chamadas por programas Lua. Isso é feito usando o seguinte protocolo: primeiro, a função a ser chamada é empilhada; então, os argumentos da função são empilhados em *ordem direta*, isso é, o primeiro argumento é empilhado primeiro. Finalmente, a função é chamada seguindo o formato

```
int lua_call (lua-state *L, int nargs, int nresults);
```

onde, nargs é o número de argumentos que foram empilhados. Para serem executados, todos os argumentos e valores da função são retirados da pilha. O resultado é ajustado para nresults e também empilhado em ordem direta, de tal forma que, após a chamada, o último resultado está no topo da pilha.

A fim de se comunicar corretamente com Lua, funções C devem seguir o seguinte protocolo, que define a forma como parâmetros e resultados são passados: a função C recebe seus argumentos de Lua pela pilha, em ordem direta. Para retornar valores para Lua, a função C empilha-os em ordem direta e retorna o número de resultados. Como uma função Lua, a função C chamada por Lua pode também retornar múltiplos valores.

O Anexo 3 ilustra o programa C++ que implementa as funcionalidades de gerência do *framework* CSBase, recebendo valores de Lua.

Para se registrar uma função C em Lua, existe a seguinte macro:

```
#define lua_register (L, n, f) (lua_pushcfunction (L, f), lua_setglobal (L, n))
/* const char *n;          */
/* Lua_CFunction f;      */
```

que recebe o nome que a função terá em Lua, e aponta para a função. Esse ponteiro deve ser do tipo lua_CFunction, que é definido como

```
typedef int (*lua_CFunction) (lua_State *L);
```

isto é, um ponteiro para uma função com resultado do tipo inteiro (*integer*), e argumento único.

A Figura 21 ilustra o registro de funções C em Lua do programa que implementa as funcionalidades de gerência do *framework* CSBase.

```
... ... ...
  lua_pushstring( L , "getcpuload\0" ) ;
  lua_pushcfunction( L , _getCpuUsage ) ;
  lua_rawset( L , -3 ) ;

  lua_pushstring( L , "getmemoryload\0" ) ;
  lua_pushcfunction( L , _getMemoryUsage ) ;
  lua_rawset( L , -3 ) ;

  lua_pushstring( L , "executecommand\0" ) ;
  lua_pushcfunction( L , _luaCreateProcess ) ;
  lua_rawset( L , -3 ) ;
... ... ...
```

Figura 21 - Registro de funções C em Lua (arquivo sgaWindows.cpp)

5.3. Fluxo de serviço do *framework* CSBase estendido

Apesar da grande evolução na obtenção de requisitos como independência de plataforma, integração, disponibilidade e simplicidade, verifica-se a ausência de um mecanismo que efetue a reserva de recursos para o SGA do *framework* CSBase. Para tanto, foi integrado em suas funcionalidades o sistema DSRT. Este novo serviço possibilita, além das funcionalidades anteriormente apresentadas, que seja efetuada a reserva dos recursos (processador) necessários, assim como as adaptações na aplicação das variações de necessidades na utilização destes recursos.

Para a execução de novas aplicações, adicionalmente deverão ser incluídos na ISI os parâmetros de reserva de processador (propriedade não funcional da aplicação ou parâmetros de QoS) (flecha 3 da Figura 22). Com base nessas informações, o SSI é capaz de comandar a execução remota das aplicações gerenciadas pelo SGA, assim como, efetuar a reserva de processador necessária (flecha 7 da Figura 22).

A nova funcionalidade inserida no SGA do *framework* CSBase, depende fortemente do estado dinâmico dos recursos computacionais do sistema, necessitando assim verificar o atendimento dos requisitos da aplicação ao longo do tempo, através de monitoração da QoS associada, reagindo a alterações relevantes através do acionamento de estratégias de adaptação apropriadas.

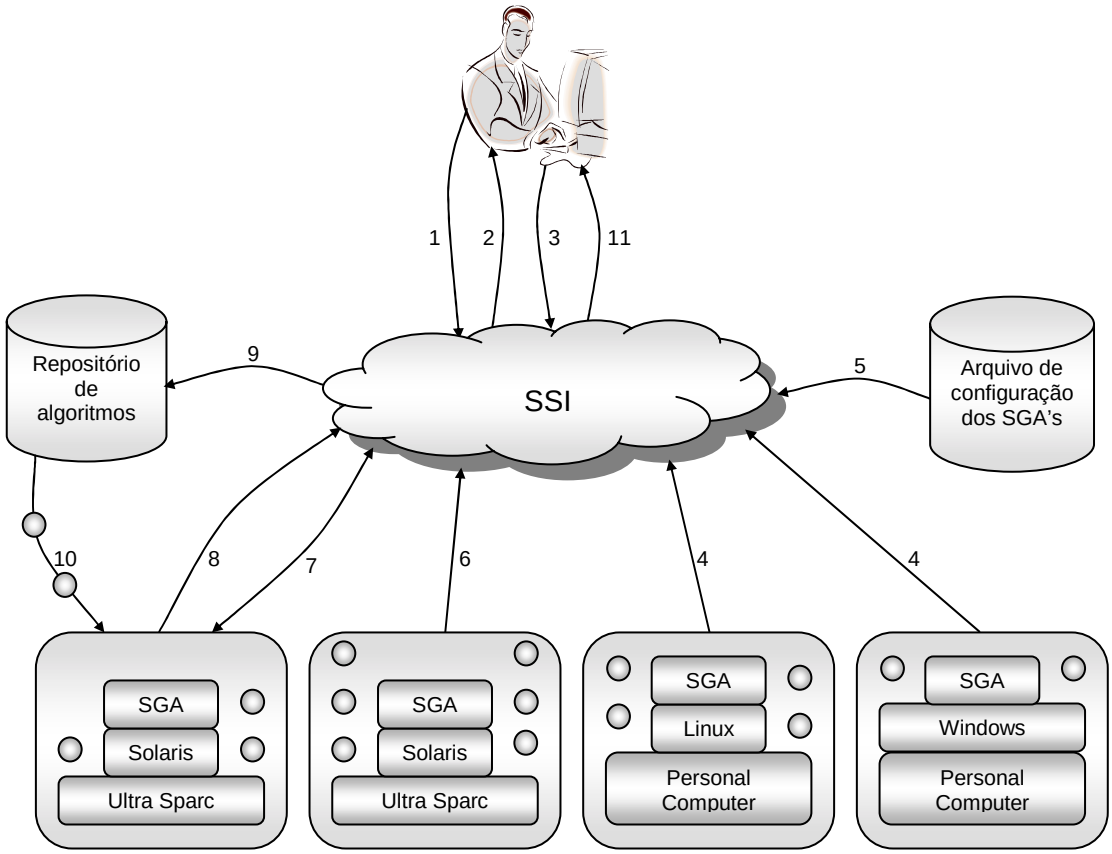


Figura 22 - Fluxo de serviço do *framework* CSBase estendido

5.4. Propriedades dos testes efetuados

Requisitos de desempenho e disponibilidade são comuns a diversos domínios de aplicações distribuídas. Em algumas aplicações, diversos clientes usam, pesadamente, um determinado tipo de serviço. Essa é uma característica fundamental no *framework* CSBase. A nova funcionalidade integrada a este sistema considera importante a limitação, sempre existente, no volume dos recursos existentes, delegando ao usuário a responsabilidade de decidir onde irá executar sua aplicação, mas, possibilitando este, uma vez decidido, reservar os recursos necessários.

Na avaliação realizada pelo usuário é importante observar uma propriedade básica para a carga submetida a uma máquina que é o tempo médio de resposta às requisições de serviço por ela atendidas. O uso dessa propriedade como uma propriedade dinâmica associada a uma oferta de serviço, permite que os clientes desse serviço selecionem as máquinas que oferecem o

melhor grau de desempenho e disponibilidade. A monitoração dessa propriedade ao longo do tempo permite que esses clientes sejam capazes, também, de detectar alterações no comportamento dessas máquinas e de alterar seu próprio comportamento em relação a essas alterações. Como a maioria dos ambientes são heterogêneos e dinâmicos, é importante o conhecimento da disponibilidade de recursos associados a essas propriedades.

Apesar de ser um projeto em contínuo aperfeiçoamento, quando se efetuou a inclusão da nova funcionalidade no SGA do *framework* CSBase, esta nova funcionalidade não foi adicionada na Interface do Sistema Integrador (ISI). Por este fato, para a realização dos testes foram necessários alguns passos adicionais aqui descritos:

1. Na inicialização do servidor SGA (daemon – arquivo sga-daemon.lua), seu IOR foi exportado para um arquivo (sga.ior) de acesso comum. Essa exportação foi necessária para possibilitar a obtenção da referência da implementação do *servant* (Figura 23). No Anexo 4 verifica-se a implementação desta operação.

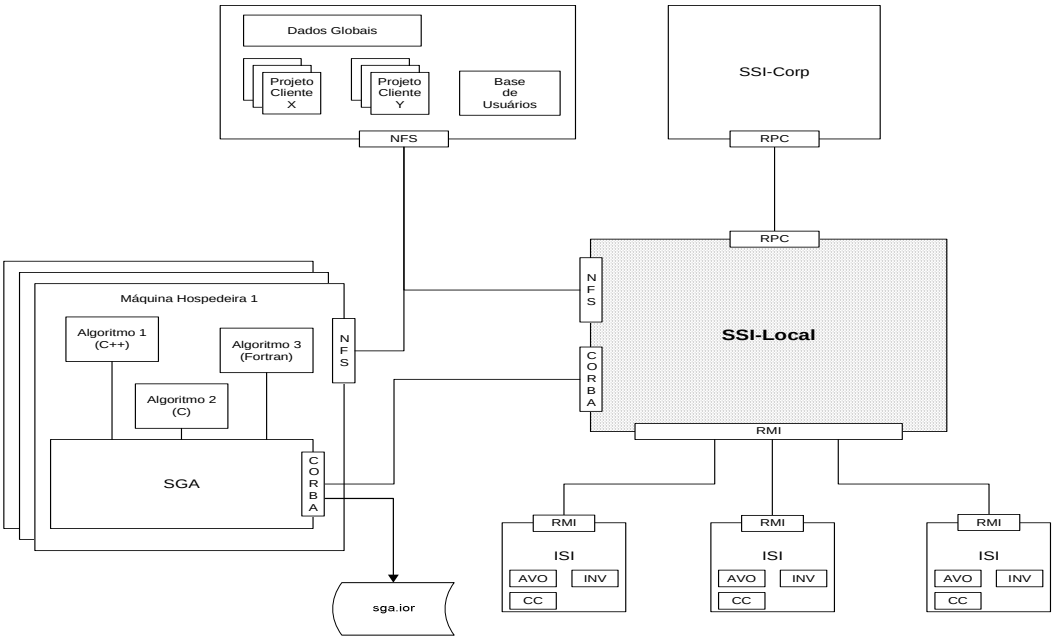


Figura 23 - Exportação do sga.ior

2. Inicialização do repositório de interfaces (IRD):

   ird -ORBIIOPAddr inet:TARSILA:8888 --ior ird.ior &

3. Novamente, devido a impossibilidade de realização dos testes via interface ISI, foi necessária a implementação de um *script* LuaORB para recuperar o IOR (sga.ior) disponibilizado no item 1, obter a referência da implementação do *servant*, criar um *proxy* genérico de acesso, e instanciar a aplicação com reserva de processador (Figura 24). Na Figura 25 verifica-se a implementação desta operação.

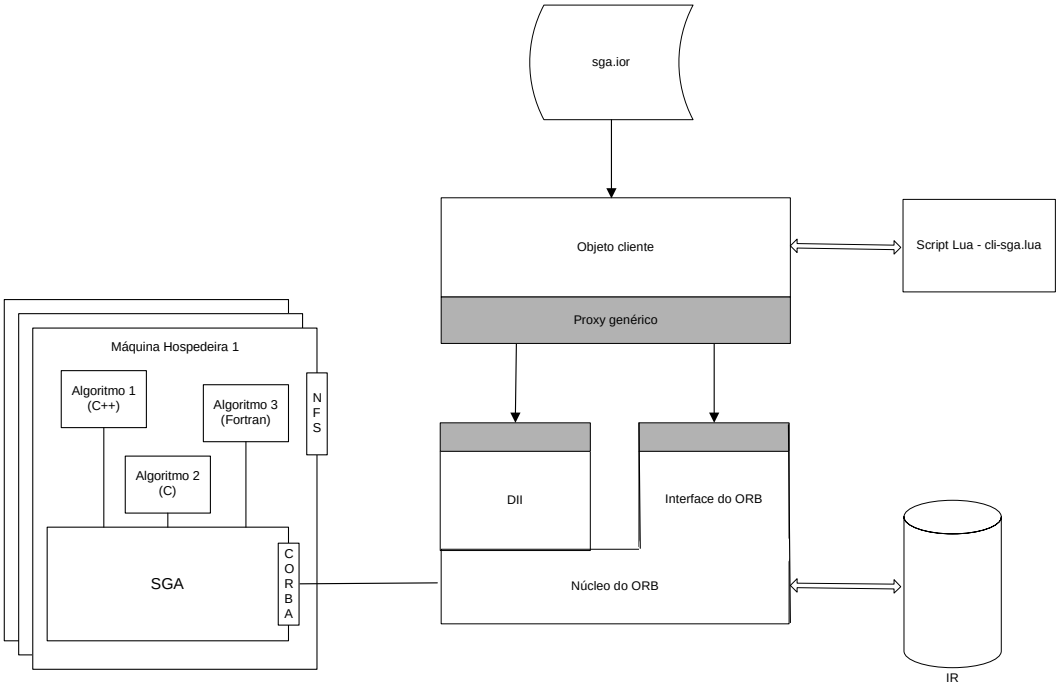


Figura 24 - Aplicação cliente de teste cli-sga.lua

```
LuaOrbCfg.setIFR(readIOR("ird.ior"))
my_proxy = lo_createproxy(readIOR("sga.ior"))
my_proxy:executeCommand("mpeg2ply.exe", "1234", "TARSILA", 10000, 0.5)
```

Figura 25 - Implementação da aplicação cliente de teste cli-sga.lua

Uma medida tradicionalmente usada em mecanismos de desempenho é a avaliação da carga submetida as máquinas, com base no número de processos na fila de prontos. É comum a diversos sistemas operacionais disponibilizarem o número de processos nessa fila, observado em intervalos diferentes. A plataforma Windows 2000 disponibiliza o utilitário Windows Management Console 1.2 versão 5.0, para que se possa verificar a execução, o desempenho e a reserva efetuados para a aplicação, em uma máquina específica.

5.5. Resultados obtidos

O objetivo dos testes não foi o de avaliar o *framework* CSBase tampouco o sistema DSRT (que pode ser encontrado em [9] [11]), mas o de verificar qual o benefício que o novo sistema integrado (CSBase estendido) trouxe para as aplicações, avaliando se realmente houve alguma melhoria na utilização dos recursos dos SGA's, pré-supondo um baixo custo (*overhead*) com o sistema DSRT [9] [11].

Todos os resultados foram obtidos reservando-se o recurso processador central, de uma máquina pertencente ao Laboratório TecGraf da PUC-Rio, Intel Pentium III 750MHz (TARSILA), 128MB (Mega Bytes) de memória RAM (*Random Access Memory*), com o sistema operacional Windows 2000 5.00.2195, *service pack* 4, e executando-se localmente um programa MPEG-2 (mpeg2ply.exe[3]) que exibe uma aplicação (4dice.mpg), a tantos quadros por segundo (fps - *frames per second*) quanto for a designação de processador central (parâmetros de QoS) a este expositor (*player*). Um exemplo típico desta variação do número de quadros por segundo pode ser verificada em um período de 50ms, que produz 20fps, e outro período de 200ms que produz 5fps.

Cada experimento contem uma ou duas instâncias da aplicação SRT periódica citada anteriormente (4dice.mpg), executando concorrentemente com um conjunto convencional de processos TS.

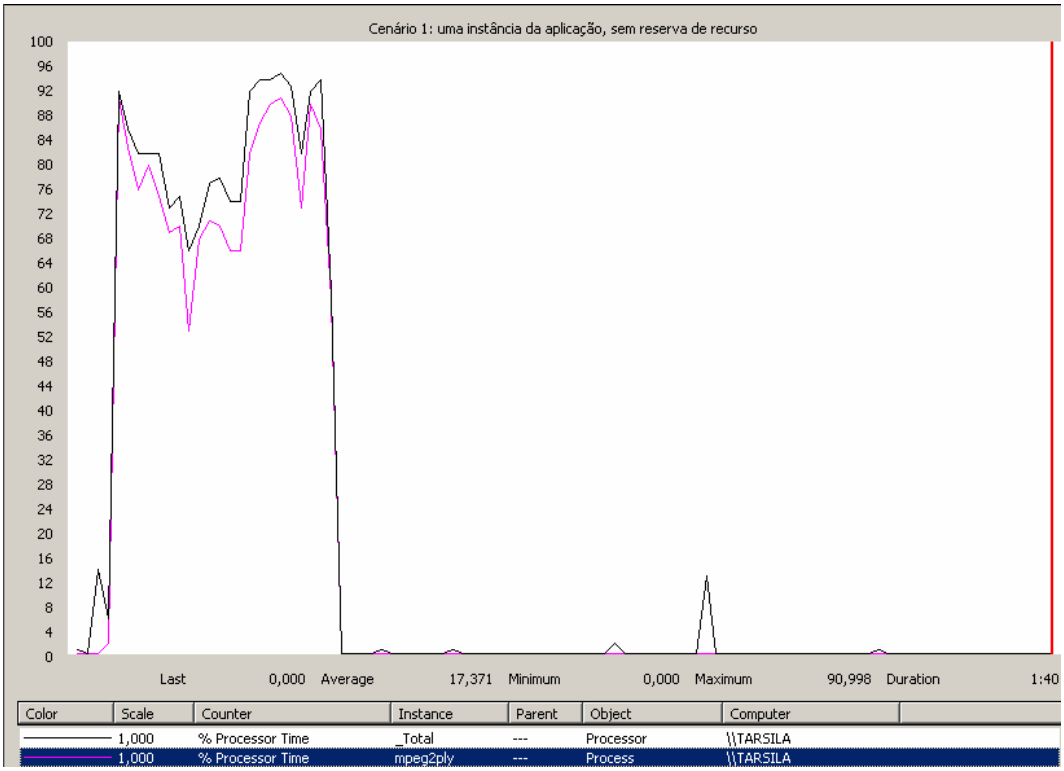
Utilizando-se a ferramenta Windows Management Console, obteve-se os seguintes resultados:

---

[3] Extensão do expositor (*player*) MPEG da Universidade de Berkeley (mpegplay), com o objetivo de forçar com que este expositor obedeça a parâmetros de QoS variáveis de exposição.

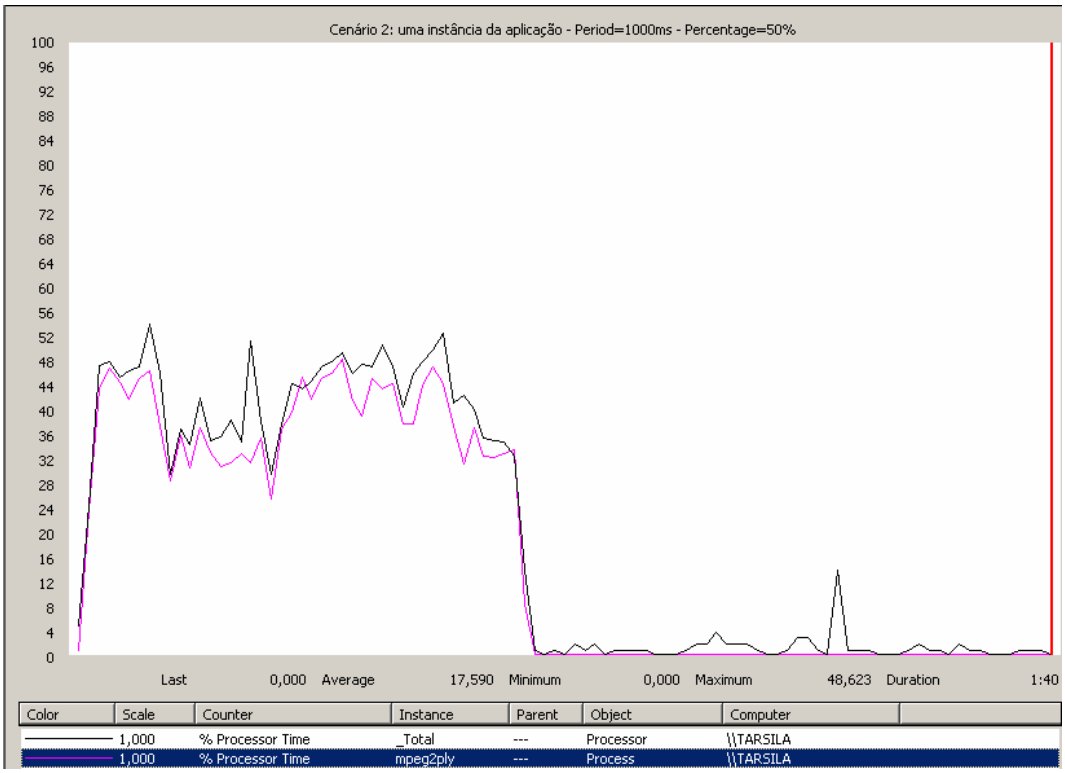
1. Cenário: uma instância da aplicação, sem reserva de recurso, ou seja, utilizando-se a classe “melhor esforço” (*best-effort*).
   Objetivo: obter-se um padrão de comportamento na execução de uma aplicação no SGA do *framework* CSBase, sem que se efetue alguma reserva de processador para esta aplicação.

   Obs: nos gráficos a seguir, a linha preta significa a quantidade total de processador utilizada, e as linhas cor de rosa e azul informam a quantidade de processador utilizada pela(s) aplicação(ões).



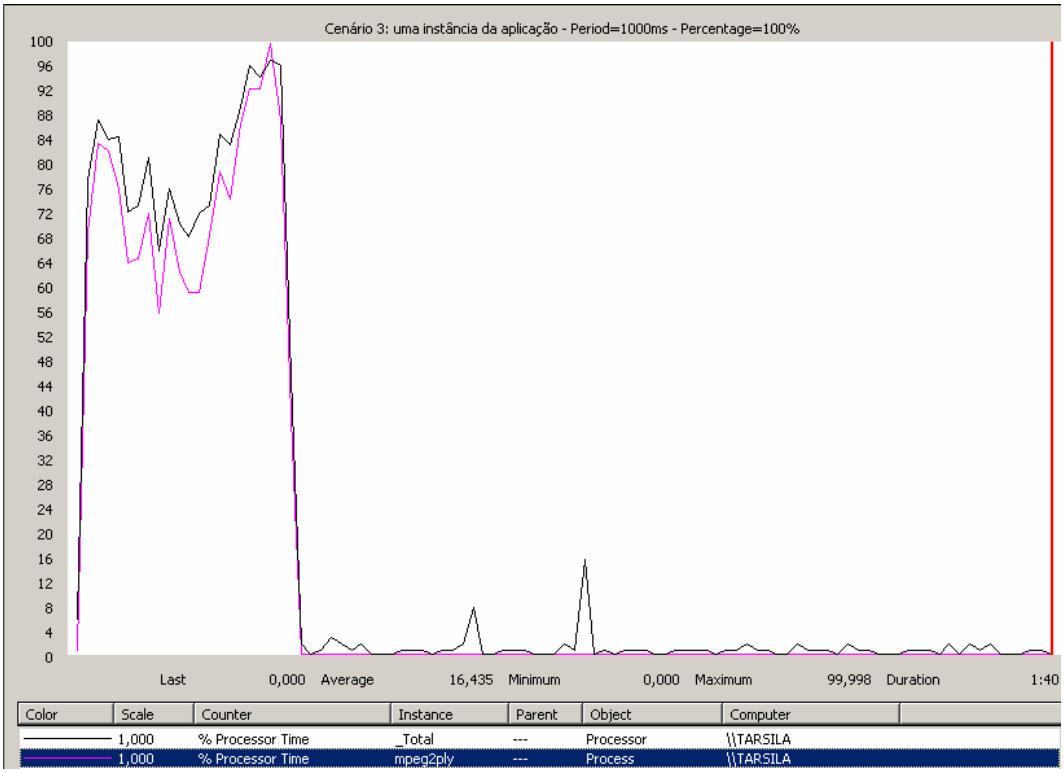
Tomando este cenário como base, verifica-se: (i) uma utilização imprevisível do processador em relação ao consumo da aplicação; (ii) uma utilização não gerenciada deste mesmo processador (a única gerência que existe é a proporcionada pelo próprio sistema operacional); e (iii) a aplicação utiliza um *quantum* aleatório de processador.

2. Cenário: uma instância da aplicação, reservando-se um período de 1000ms, e um percentual de utilização do processador de 50%.
   Objetivo: obter-se a avaliação de uma aplicação executada pelo SGA do *framework* CSBase, efetuando-se uma reserva limitada de processador para esta aplicação. Com este cenário pode-se verificar a nova capacidade do SGA, como agente limitador de uso da capacidade do processador para aplicações específicas.



Nesse cenário percebe-se: (i) uma utilização previsível do processador em relação ao consumo da aplicação; (ii) uma utilização gerenciada deste mesmo processador, com essa gerência sendo proporcionada pelo novo mecanismo de gerência de recursos ofertado pelo SGA do CSBase estendido; e (iii) a aplicação passou a utilizar o *quantum* de processador que lhe foi designado.

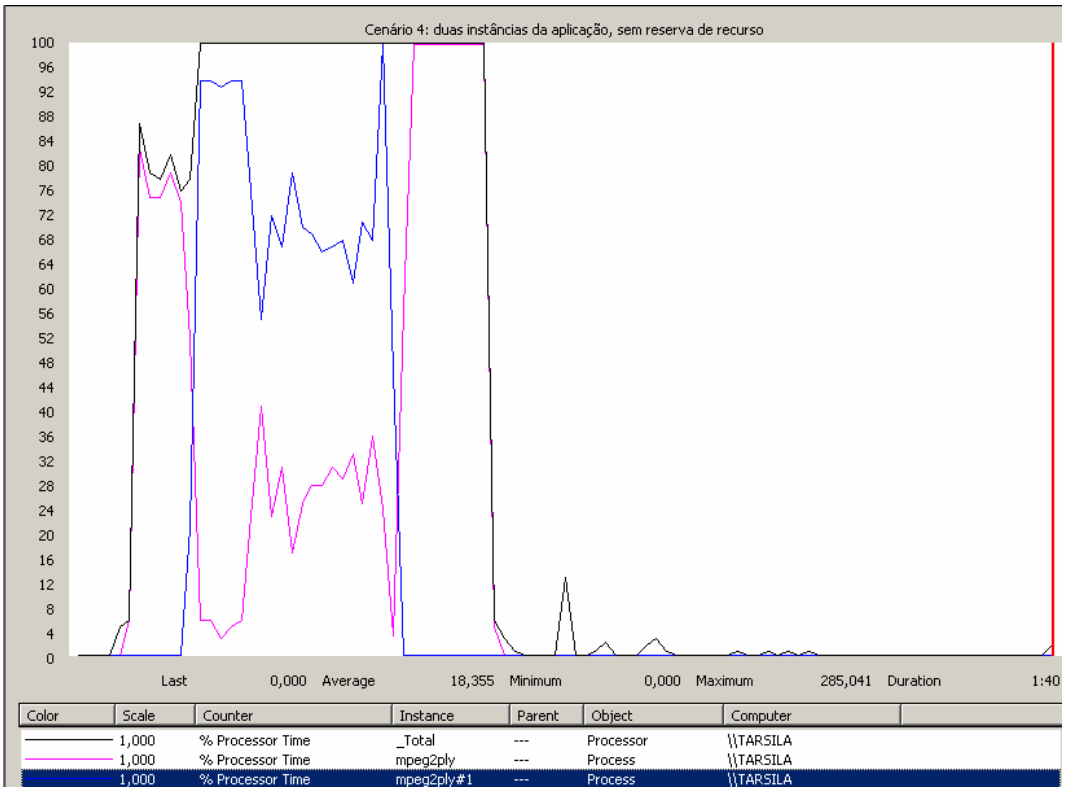
3. Cenário: uma instância da aplicação, reservando-se um período de 1000ms, e um percentual de utilização do processador de 100%.
   Objetivo: obter-se a avaliação de uma aplicação executada pelo SGA do *framework* CSBase, efetuando-se uma reserva de 100% de processador para esta aplicação, ou seja, privilegiando-se o máximo possível esta aplicação, no que concerne a utilização do processador. Com este cenário pode-se verificar outra nova capacidade do SGA em privilegiar ao máximo o uso da capacidade do processador para aplicações específicas.



Analogamente ao Cenário 2, percebe-se: (i) uma utilização previsível do processador em relação ao consumo da aplicação; (ii) uma utilização gerenciada deste mesmo processador, com essa gerência sendo proporcionada pelo novo mecanismo de gerência de recursos ofertado pelo SGA do CSBase; e (iii) a aplicação passou a utilizar o *quantum* de processador que lhe foi designado.

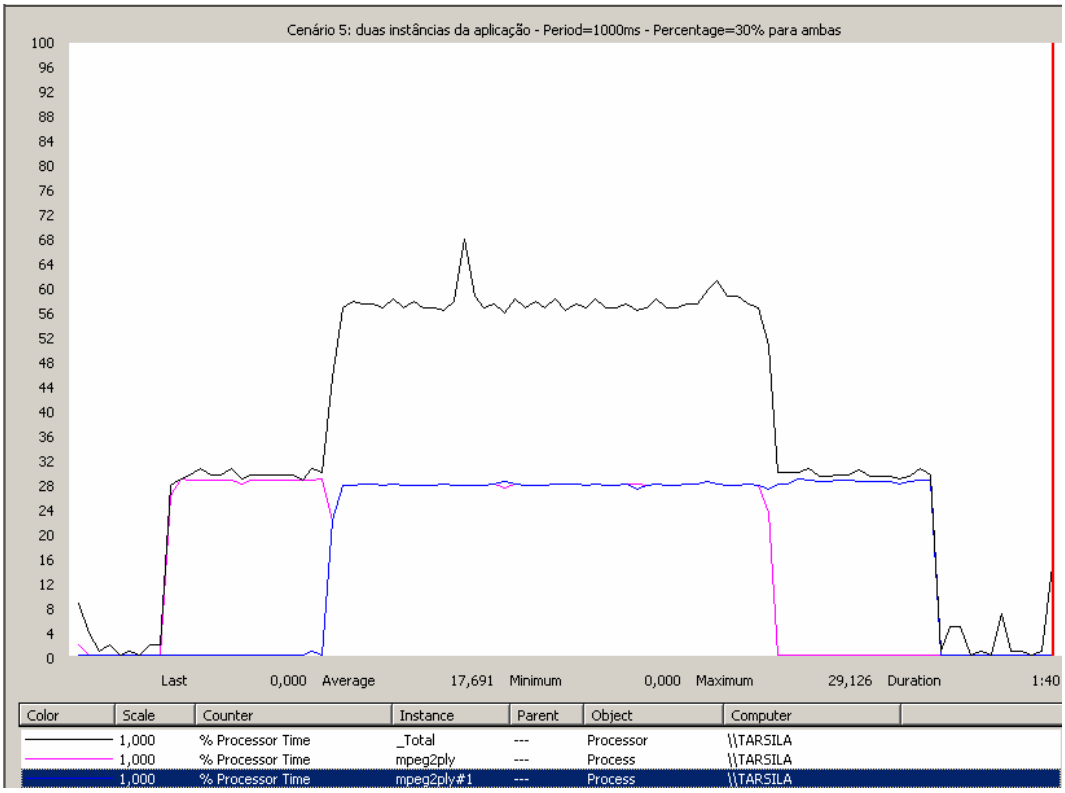
A semelhança aparente com o gráfico do Cenário 1 deve-se ao fato de, na instanciação deste Cenário 1, não haver nenhum outro processo concorrendo com este para o uso do processador central.

4. Cenário: duas instâncias da aplicação, sem reserva de recurso, ou seja, utilizando-se a classe “melhor esforço” para ambas.
   Objetivo: obter-se um padrão de comportamento na execução de duas aplicações pelo SGA no *framework* CSBase, sem efetuar-se alguma reserva de processador para estas aplicações.



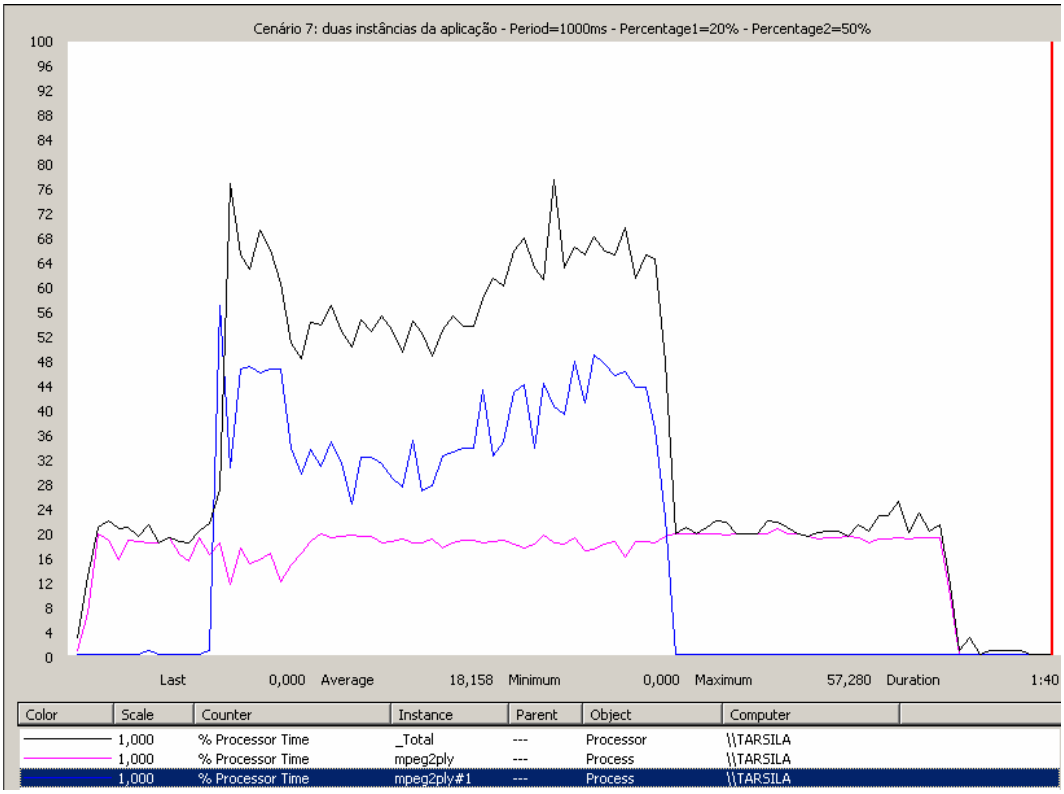
Neste cenário percebe-se: (i) uma utilização imprevisível do processador em relação ao consumo das aplicações; (ii) uma utilização não gerenciada deste mesmo processador; (iii) as aplicações utilizam um *quantum* aleatório de processador; e (iv) é impossível estabelecer uma relação temporal entre as aplicações (supondo as duas com o mesmo tempo de execução).

5. Cenário: duas instâncias da aplicação, reservando-se um mesmo período de 1000ms para ambas, e um mesmo percentual de utilização do processador de 30%.
   Objetivo: obter-se a avaliação de duas aplicações executadas pelo SGA do *framework* CSBase, efetuando-se uma reserva limitada (mas igual) de processador para ambas aplicações. Com este cenário pode-se verificar a nova capacidade do SGA, como agente limitador de uso da capacidade do processador para aplicações específicas.



Neste cenário verifica-se: (i) uma utilização previsível do processador em relação ao consumo das aplicações; (ii) uma utilização gerenciada deste mesmo processador; (iii) as aplicações passaram a utilizar o *quantum* de processador que lhe foi designado; (iv) é possível o estabelecimento de uma relação temporal entre as aplicações (supondo as duas com o mesmo tempo de execução); e (v) esta relação temporal é totalmente previsível.

6. Cenário: duas instâncias da aplicação, reservando-se um mesmo período de 1000ms para ambas, e um percentual de utilização do processador de 20% para a primeira e de 50% para a segunda.
   Objetivo: obter-se a avaliação de duas aplicações executadas pelo SGA no *framework* CSBase, efetuando-se uma reserva limitada (mas diferente) de processador para estas aplicações. Com este cenário pode-se verificar a nova capacidade do SGA, como agente gerenciador de uso da capacidade do processador para aplicações específicas.



Neste cenário percebe-se: (i) uma utilização previsível do processador em relação ao consumo das aplicações; (ii) uma utilização gerenciada deste mesmo processador; (iii) as aplicações passaram a utilizar o *quantum* de processador que lhe foi designado; (iv) é possível o estabelecimento de uma relação temporal entre as aplicações (supondo as duas com o mesmo tempo de execução); e (v) esta relação temporal é totalmente previsível.