O Upstart substitui o Sys V Init e acelera a inicialização do sistema

Inicialização em um pulo

O lento processo de inicialização do Linux vem incomodando os usuários há alguns anos. Agora, o projeto Upstart oferece uma nova abordagem para essa questão.

por Nico Dietrich e Dirk von Suchodoletz

A história do Linux inclui diversas tentativas de solucionar o problema do longo processo de inicialização do sistema. Isso não é uma surpresa, pois longas maratonas como essa incomodam todos os usuários. O projeto de inicialização legado do *Unix System V* foi revolucionário ao ser lançado, mas se tornou um peso para as distribuições modernas.

Embora tenham surgido vários truques para acelerar esse procedimento, a maioria se mostrou impraticável, e grande parte dos carregadores de serviço "turbinados" criados pelos gurus não são utilizáveis por usuários normais. Uma nova ferramenta, o *Upstart* [1], apresenta um *daemon* de *init* genérico que impulsiona vários desenvolvimentos em sistemas Linux modernos. O Upstart, com suas raízes no Unix, evita de forma inteligente longas esperas, o que reduz o intervalo de inicialização a um mínimo. A longo prazo, o plano é substituir os serviços genéricos de segundo plano, como os *daemons at*, *cron* e outros, pelo Upstart. O *Ubuntu* 6.10 (*Edgy Eft*) demonstra os primeiros efeitos desse promissor software.

Tudo começa com o init

A maioria dos sistemas relacionados ao Unix compartilha o conceito do Init. O processo chama o kernel e atribui o *ID* 1 de processo do kernel. Essa seqüência está gravada no próprio kernel Linux (`/usr/src/linux/init/main.c`). O Init é incumbido de iniciar todos os outros processos do espaço do usuário e, conseqüentemente, iniciar a máquina (**quadro 1**). O processo e seus scripts auxiliares carregam módulos do kernel, verificam e montam os sistemas de arquivos, estabelecem conexões de rede, iniciam servidores e chamam o gerenciador gráfico de *login*. O Init deve iniciar os serviços em uma ordem que faça sentido. Por exemplo, é errado configurar a hora do sistema consultando um servidor de hora na rede até que a máquina tenha, de fato, acesso à rede. Para isso, o Init primeiro tem que inicializar o hardware de rede e estabelecer ao menos uma conexão para acesso externo.

O número de serviços e agentes de segundo plano cresceu ao longo dos anos, o que bagunçou o processo de inicialização. Por outro lado, o uso em desktops, como é típico no caso do Ubuntu, necessita de um sistema de configuração dinâmico. Dispositivos móveis também dificultam consideravelmente a vida do *Sys V Init*. Máquinas em movimento demandam uma abordagem *ad hoc* para estabelecer conexões de rede, assim como uma forma de configurar o hardware dinamicamente.

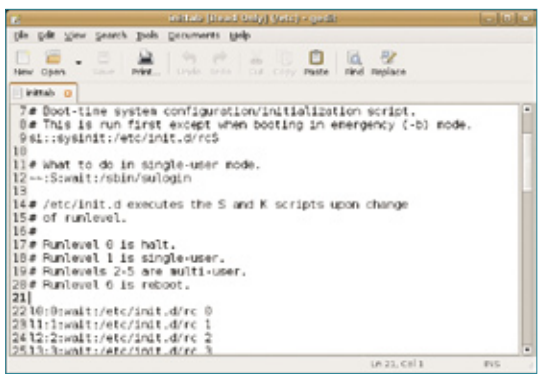

Figura 1 Esse `inittab` típico define os *runlevels* de 0 a 6.

Quadro 1: Sys V Init

As primeiras versões do Unix usavam um script simples de *shell* para configurar a máquina e executar serviços. O projeto por trás do `/etc/rc` da família *BSD*, por exemplo, é convincentemente simples, porém possui uma importante desvantagem: Integrar softwares de terceiros, ou extensões personalizadas, significava modificar o script de shell. Infelizmente, alterar esse código é muito perigoso – um simples erro poderia deixar o sistema em estado inutilizável.

Em muitos casos, é necessário mais de um comando para iniciar um serviço, com os detalhes variando de acordo com o ambiente. Por exemplo, o servidor *DHCP ISC* pode ser configurado para escutar em interfaces *ethernet* específicas, em vez de todas as portas. Para eliminar a necessidade de os administradores modificarem o script de inicialização com esse fim, os *daemons* costumam trazer arquivos de configuração que analisam o script. Isso permite que o administrador atualize o script de inicialização sem pôr em perigo a configuração local.

O *Sys V Init* usa uma abordagem bem mais flexível, e também mais complexa, introduzindo o conceito de *runlevels* que definem estados específicos da máquina, com base nos processos em execução neles. No total, oito runlevels são possíveis, mas não obrigatórios, em qualquer sistema. Três deles desempenham tarefas definidas: *Halt* (0), Modo *single user* (1) e *Reboot* (6).

O arquivo `/etc/inittab` especifica quais *runlevels* existem e define aquele em que o sistema entrará ao iniciar (**figura 1**).

O projeto do Sys V presume que o sistema usará um pequeno número de estados definidos, tais como ausência de rede, presença de rede, presença do *X11* e assim por diante. Os administradores podem alternar para outro runlevel com o comando `init runlevel`.

Outra vantagem dessa abordagem está nos scripts separados para cada serviço ou tarefa de configuração. Por exemplo, chamar `/etc/init.d/dhcpd restart` permite que se reinicie o servidor DHCP sem afetar outros serviços. A idéia de usar um conjunto de *links* simbólicos para determinar o escopo e a ordem dos scripts usados em cada runlevel também é interessante.

Exemplo 1: Definições de tarefas

```
01 # /etc/event.d/rc2
02 # Script do Upstart de compatibilidade com o runlevel 2
03 # Esta tarefa roda os antigos scripts SysV
04
05 start on runlevel-2
06
07 stop on shutdown
08 stop on runlevel-3
09 stop on runlevel-4
10 stop on runlevel-5
11
12 script
13    set $(runlevel --set 2    || true)
14    if [ "$1" != "unknown" ]; then
15       PREVLEVEL=$1
16       RUNLEVEL=$2
17       export PREVLEVEL   RUNLEVEL
18    fi
19
20    exec /etc/init.d/rc 2
21 end script
```

Para lidar com essas questões específicas, alguns programadores desenvolveram diversas ferramentas: o *acpid* e o *apmd* para gerenciamento de energia, o gerenciador de dispositivos do *HAL* para montagem dinâmica de dispositivos, e o *Resource Manager* (gerenciador de recursos) para a atribuição dinâmica de permissões de dispositivos para usuários de interfaces gráficas. Cada um desses sistemas implementa sua própria lógica de configuração, e os administradores têm que se familiarizar com essas lógicas para conseguir executar determinada tarefa no momento certo.

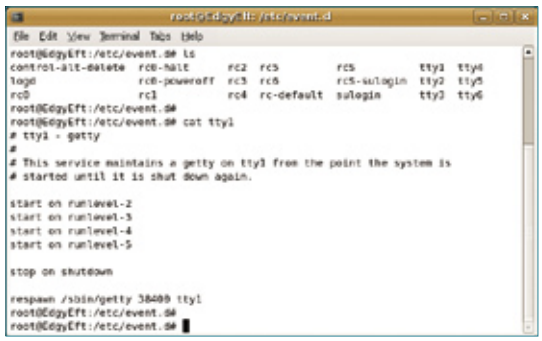

Figura 2 O *Edgy Eft* armazena os arquivos que definem as tarefas de eventos típicos do `inittab` legado em `/etc/event.d`.

Nem todos os processos e serviços limitam-se a iniciar ou desligar uma máquina. Por exemplo, há alguns serviços especiais, como o cron e o daemon at, que iniciam outros processos num momento específico. Eles não são vinculados ao sistema de *runlevels* de forma alguma, apesar de possuírem uma lógica subjacente. Esse é outro alvo de mudanças do Upstart [3].

Problemas de projeto

Antes de decidirem desenvolver um novo sistema, os programadores do Ubuntu primeiro investigaram alternativas comtemporâneas do System V [2]. Nenhum dos projetos que viram satisfez suas expectativas ou estava disponível sob uma licença aceitável.

Quando começaram a pensar em um novo projeto, precisaram optar entre uma abordagem orientada ao alvo ou ao resultado, em relação à inicialização do sistema. A escolha orientada ao alvo significaria definir os serviços que deveriam estar iniciados ao final da seqüência de inicialização (*KDM*, daemon *SSH* etc.).

Nesse caso, seria necessário investigar cada serviço e determinar de quais outros serviços ele depende. Baseado nessas dependências, o sistema de inicialização deveria deduzir uma seqüência de inicialização sensata. Essa é exatamente a abordagem empregada pelo *Gentoo Linux*, com seu sistema *depend* (**quadro 2**); o *Suse* também segue esse princípio com uma versão modificada do Sys V Init (**quadro 3**).

No outro canto do ringue, havia os eventos. Em vez de formular dependências, que um script provavelmente precisaria tratar durante a inicialização do sistema, um sistema baseado em eventos não executaria um script até que um conjunto específico de pré-condições fossem satisfeitas. Por exemplo, não faria sentido chamar um cliente *NFS* até que a infraestrutura para isso estivesse disponível. O sistema que o Ubuntu escolheu também aceita condições mais complexas, tais como "configuração de rede completada", "Apache rodando" ou (futuramente) "Disco USB conectado".

Horizonte de eventos

Eventos são, basicamente, cadeias de caracteres. Os desenvolvedores do Upstart os dividem em três classes:

- Eventos marginais simples, como "o sistema está sendo inicializado" ou "usuário pressionou um botão".
- Eventos de nível possuem um parâmetro adicional, como o status da interface de rede. Os serviços e tarefas podem ser executados tanto para eventos de nível quanto no momento em que um parâmetro atingir um valor específico.
- Eventos temporais ocorrem após um intervalo específico ou em um dado momento.

Os desenvolvedores mantiveram o mandamento do Código Aberto: liberar logo, liberar com freqüência. Assim, o código foi lançado ao público em um estágio bem inicial, e os auto-confiantes desenvolvedores apresentaram um sistema em funcionamen-

Exemplo 2: Controle de serviços

```
01 root@EdgyEft:~# start simple-server
02 simple-server (start) running, process 6507 active
03 root@EdgyEft:~# stop simple-server
04 simple-server (stop) running, process 6507 killed
05 root@EdgyEft:~# status simple-server
06 simple-server (stop) waiting
07 root@EdgyEft:~# start simple-server
08 simple-server (start) running, process 6517 active
09 root@EdgyEft:~# status simple-server
10 simple-server (start) running, process 6517 active
```

Quadro 2: Gentoo

Como uma das distribuições baseadas no Kernel Linux mais recentes, o *Gentoo* resolveu o problema da organização de scripts de *runlevel* de uma forma especial. Ele não utiliza scripts de *Bash* simples, mas um interpretador separado: `/sbin/runscript`. Um exemplo de estrutura típica é esse:

```
#!/sbin/runscript

opts="depend start stop restart"
depend() {
  # Dependencias e condicoes
}
start() {
  # Comandos para iniciar servicos, # incluindo preparativos
}
stop() {
  # Comandos para parar servicos, # e acoes de limpeza
}
restart() {
  # Reiniciar como servico
}
```

O texto após `opts` lista todas as funções fornecidas pelo script de runlevel. Se for necessário acrescentar suas próprias funções, basta adicioná-las à lista e criar um bloco de função com o mesmo nome, no próprio script. Enquanto as seções `start`, `stop` e `restart` mantêm o projeto tradicional, coisas mais interessantes acontecem em `depend`. Um serviço depende de outros serviços ou configurações preparatórias por um lado; mas, por outro, ele pode fornecer funções requeridas por outros serviços:

- `need serviço`: Depende do serviço.
- `use serviço`: Usa o serviço.
- `provide funcionalidade`: Fornece uma funcionalidade específica.
- `before serviço`: Deve ser iniciado antes de outro serviço.
- `after serviço`: Deve iniciar após o serviço especificado.

O Gentoo também suporta serviços virtuais, como *net*, pois há vários tipos de rede (*ethernet, modem, WLAN*). Isso também se aplica a servidores de email (*mta*). O script de inicialização pode até mesmo determinar dinamicamente as dependências, como mostra o `/etc/init.d/syslog-ng`:

```
case $(sed 's/#.*//' /etc/syslog-ng/syslog-ng.conf) in
<b>source*tcp</b>|<b>source*udp</b>|<b>destination*tcp</b>|<b>destination*udp</b>)
➥need net ;;
esac
```

Contanto que a mudança não entre em conflito com dependências existentes, o administrador tem a possibilidade de alterar a ordem na qual os serviços são iniciados, usando `before` ou `after`.

to, rodando o Edgy Eft, para demonstrar o ponto que já haviam atingido. Seu objetivo é coletar o máximo possível de retorno de outros desenvolvedores que trabalhem em outras distribuições Linux.

Entretanto, isso também significa que as especificações podem mudar nos próximos meses. Os exemplos descritos neste artigo referem-se à versão 0.3, do início de dezembro de 2006.

Essa versão substituiu o processo de init já existente. No entanto, nem todos os scripts de inicialização foram modificados para usar o mecanismo de eventos, e há planos de usar outros programas, como o *Udev* e os daemons ACPI e APM, como fontes de eventos.

Estado atual

A versão atual do Ubuntu tem inicialização bastante rápida, embora não se veja o que acontece por baixo dos panos na configuração padrão, pois a tela de *bootsplash* esconde as informações. Apesar de os usuários normais não se queixarem disso, leva algum tempo para os administradores se habituarem. Mesmo que se elimine o splash colorido (apagando o termo `splash` da linha de comando do kernel no *Grub*), não haverá muitas mensagens. Se for interessante ver mais, pode-se apenas apagar a entrada `quiet`.

Num primeiro olhar, as mudanças sob o capô ainda estão ocultas. Os comandos `man init` e `man telinit` informam que o sistema de runlevels tem um novo mecanismo. Outro indicativo da mudança é a ausência do arquivo `/etc/inittab`. Os scripts auxiliares em `/etc/init.d/` também existem, ainda, pois o Ubuntu utiliza o Upstart em modo de compatibilidade. O plano no médio prazo é permitir que o `/etc/event.d/` cuide das definições de ta-

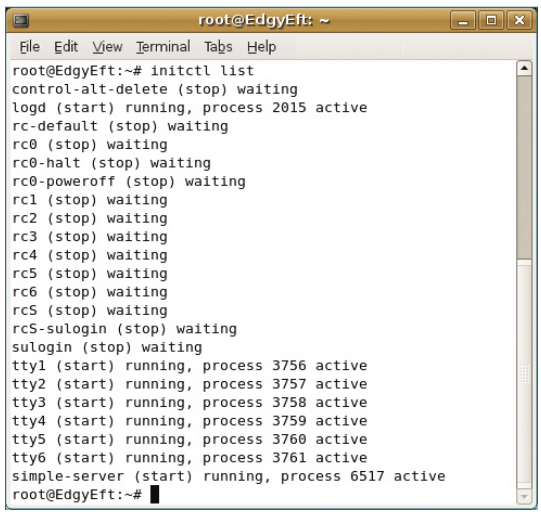

Figura 3 O comando `initctl list` fornece um panorama do estado do sistema.

refas, o que se resume a arquivos simples e não executáveis, como o do **exemplo 1**. O exemplo usa o caminho fácil, simplesmente chamando os velhos scripts de inicialização do runlevel 2 (`linha 20`).

Como se vê na `linha 5`, queremos que o script rode sempre que o evento *runlevel-2* ocorrer. Ele termina se os eventos *shutdown* ou *runlevel-3* a *runlevel-5* ocorrerem (`linhas 7 a 10`). No futuro, uma semântica mais complexa suportará condições com operadores lógicos, e será capaz de passar parâmetros a scripts de eventos, caso seja necessário. Esses arquivos cumprem o mesmo papel das entradas no antigo `/etc/inittab`. É por isso que o Edgy Eft possui tanto o `rc2` quanto os arquivos mostrados no alto da **figura 2**.

Tarefas compatíveis com o Upstart

Há duas formas de definir tarefas. O método simples usa a abordagem `exec /caminho/do/programa -O --parâmetro_opcional`. Isso funciona exatamente como na *shell*. O Upstart na realidade usa uma shell para lidar com aspas duplas e simples, além de `$`. Se a definição da tarefa contiver mais do que uma simples linha de comando, o script de shell pode residir entre os termos `script` e `end script` (**exemplo 1**, **linhas 12** a **21**).

Existem duas variações desse tema de scripts: `start script` e `stop script`. O `start script` faz o que o serviço necessita, como a criação de diretórios ou a verificação de privilégios de acesso. O `stop script` limpa tudo após o término do serviço.

Auto-executável

Usando um servidor simples como exemplo, vejamos os passos envolvidos na criação de um script personalizado para o Upstart. O servidor não precisa fazer nada além de continuar em funcionamento. A seguinte seção se baseia em um comando de duas linhas, em `/usr/local/bin/simpleserver.sh`:

```
#/bin/sh
while true ; do sleep 1 ; done
```

Vamos chamar o script de eventos para esse serviço de `/etc/event.d/simple-server`. Se quisermos suportar apenas a execução manual do serviço, só precisaremos de uma única linha no script de eventos para iniciar o servidor:

```
exec /usr/local/bin/simpleserver.sh.sh
```

Quadro 3: Suse Linux

Enquanto o *Sys V Init* tradicional segue uma abordagem estritamente linear, versões mais recentes do *Suse Linux* (10.0 e posteriores) suportam a paralelização de chamadas de scripts de inicialização. Os administradores podem ativar esse recurso no arquivo `/etc/sysconfig/boot`, especificando o valor `yes` na variável `RUN_PARALLEL`. Isso modifica a seqüência legada definida por `S00script1` a `S99script25`.

Em vez disso, as dependências de `.depend.boot`, `.depend.start` e `.depend.stop` são aplicadas. Se o administrador acrescentar um script simples, digamos, `S12nbd-server`, a `rc3.d`, criando um *link* da forma tradicional, o sistema vai simplesmente ignorar a modificação.

O comando `insserv` cuida dessa tarefa avaliando o cabeçalho do arquivo para garantir a resolução correta das dependências:

```
### BEGIN INIT INFO

Provides: nbd-server
Required-Start: $network
Should-Start: $syslog
Required-Stop:
Default-Start: 3 5
Default-Stop:  0 1 2 6
Description: Start Network Blockdevice Daemon

### END INIT INFO
```

Isso esconde do usuário boa parte da complexidade. Porém, não oferece grandes avanços em relação à velocidade. Quando testamos a inicialização no estilo do *Suse* em nossa máquina de testes – a qual sabemos que possui um disco rígido deveras lento – a inicialização paralela levou pouco mais de um minuto, o que está bem próximo do valor de 70 segundos da forma legada. É possível, no entanto, ver algumas pistas do início paralelo dos serviços: a saída na tela fica misturada.

Exemplo 3: Inicialização com Upstart

```
01 [...]
02 Dec 3 18:44:59 rc2: * Starting deferred execution scheduler atd...     [ ok ]
03 Dec 3 18:44:59 rc2: * Starting periodic command scheduler...           [ ok ]
04 Dec 3 18:44:59 rc2: * Enabling additional executable binary formats... [ ok ]
05 Dec 3 18:44:59 rc2: * Checking battery state...                        [ ok ]
06 Dec 3 18:44:59 rc2: * Running local boot scripts (/etc/rc.local)      [ ok ]
```

Para iniciar e parar os serviços, incluindo o que acabamos de definir, ainda precisaremos dos comandos `start` e `stop`, além de um novo, chamado `status`.

Um simples `start simple-server` traz o serviço à vida. Para verificar se o comando funcionou, use `initctl list` ou `status simple-server`. Um `stop simple-server` terminará o serviço (**exemplo 2**).

Se tudo estiver funcionando, a maioria dos usuários não terá interesse em ver as mensagens do sistema. Contudo, informações dos registros podem ser úteis, especialmente se tivermos acabado de modificar o sistema. Se você preferir não emitir qualquer mensagem durante a inicialização, ainda será possível verificá-las depois, obviamente.

De uma forma geral, a saída do script do Upstart é passada para o *logd* incluído no pacote, e o daemon os deixa em `/var/log/boot` (**exemplo 3**). O comando `initctl list` fornece outra forma de saída de depuração (**figura 3**).

Upstart no Debian

Como o Ubuntu se baseia no *Debian*, a probabilidade de se conseguir acelerar o Debian, graças às modificações do Ubuntu, são bem altas. Se você estiver preparado para aceitar o risco, é possível optar entre substituir completamente o Sys V Init já existente e usar o Upstart em paralelo com ele.

Os passos para implementar o plano A (usar o Upstart em total substituição ao sistema legado) no Debian Unstable são bem simples – os desenvolvedores da distribuição já terminaram os preparativos separando as *sysvinit-utilis* do pacote *sysvinit*. Isso significa que é fácil substituir o pacote *sysvinit* pelo Upstart, e então simplesmente manter os scripts antigos.

Existe um pacote para o Upstart no repositório *Experimental* [2] do Debian. Para usá-lo, acrescente a seguinte entrada ao arquivo `/etc/apt/sources.list`:

```
deb http://ftp.de.debian.org/debian/
➥experimental main
```

Exemplo 4: Configuração do Grub

```
01 # /boot/grub/menu.lst
02 [...]
03 title Ubuntu, kernel 2.6.17-10-generic
04 root (hd0,0)
05 kernel /boot/vmlinuz-2.6.17-10-generic root=/dev/hdb1 ro quiet splash init=/opt/upstart/sbin/init
06 initrd /boot/initrd.img-2.6.17-10-generic
07 boot
08 [...]
```

Em seguida, remova o pacote legado *sysvinit* e instale no lugar os equivalentes do Upstart:

```
apt-get install upstart upstart-compat-sysv
```

Como o *sysvinit* está marcado como *required* (obrigatório), o gerenciador de pacotes vai esperar até que o usuário digite *Sim*. A próxima dificuldade ocorrerá na atualização. O comando `apt-get dist-upgrade` irá remover os pacotes do Upstart recém-instalados e colocar no lugar o *sysvinit*.

Se essa for sua intenção – restaurar o uso do Sys V Init – você pode simplesmente usar o comando `apt-get install sysvinit`, que o resultado será o mesmo.

Auto-administração

Se você decidir compilar o Upstart, será necessário remover manualmente o pacote `sysvinit`. Se isso não for feito, o `make install` irá sobrescrever os binários centrais, e o gerenciador de pacotes ignorará as mudanças, ou, pior ainda, passará a acreditar que seu sistema foi corrompido. Dito isso, é fácil compilar e instalar o Upstart a partir do código-fonte [1]:

```
./ configure --prefix=/usr --exec-prefix=/
➥--sysconfdir=/etc
make
make install
```

Após terminar esses passos, o sistema pedirá seus scripts de inicialização.

Para começar, é necessário baixar o *tarball* `example-jobs-2.tar.gz` a partir do diretório `/download/[1]` e descompactá-lo em `/etc/event.d/`.

Mundos paralelos

Se você desejar evitar que um erro de instalação do Upstart estrague seu sistema Sys V Init, é possível instalar o Upstart lado a lado com o Sys V. Para isso, siga os mesmos passos requeridos para instalação do Upstart solitário, mas mantenha o pacote *sysvinit* e certifique-se de que o novo Init seja colocado em `/opt/upstart`:

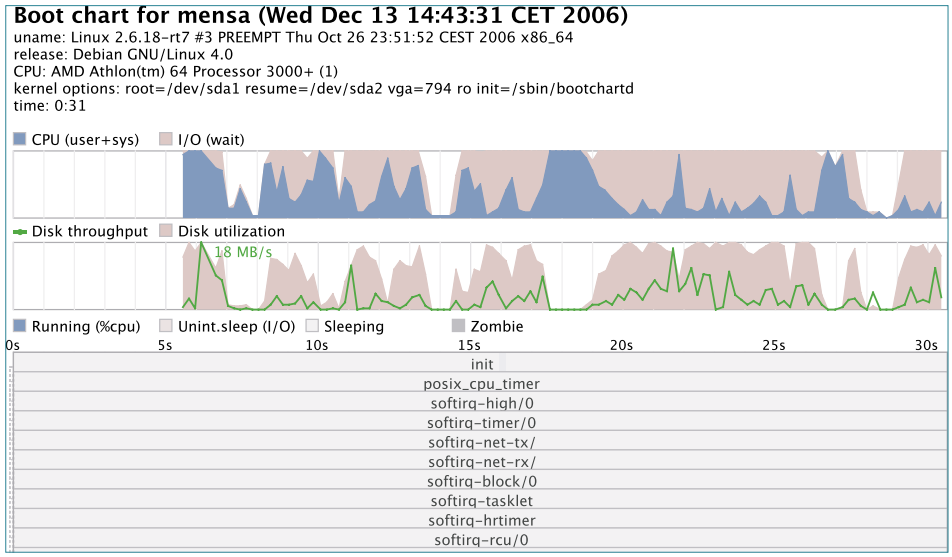

Figura 4 Esta análise do *Bootchart* mostra um sistema *Debian GNU/Linux* iniciando com o procedimento *init* legado.

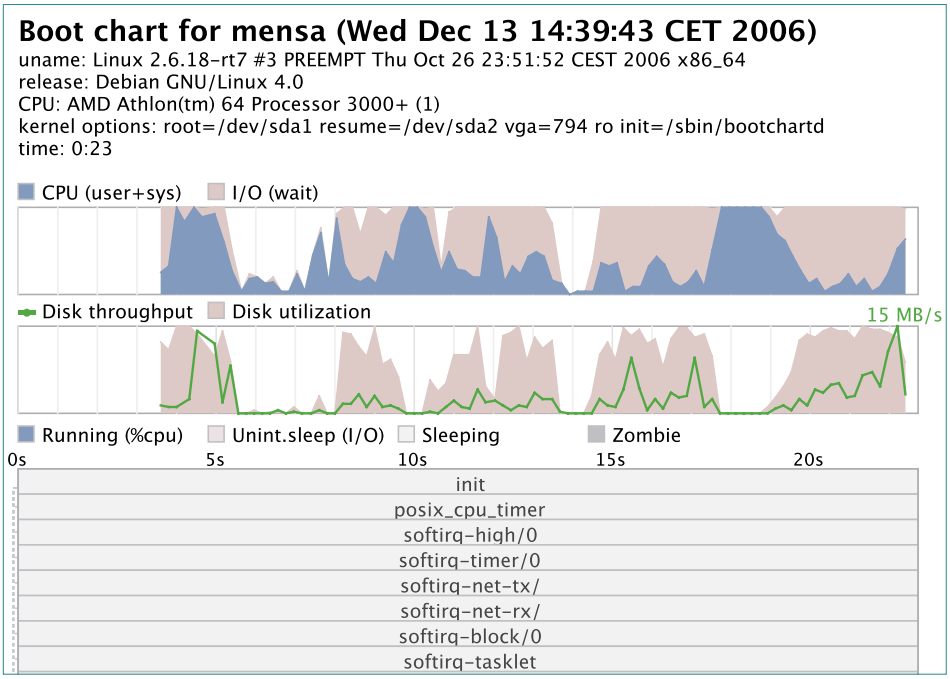

Figura 5 Re-executar o *Sys V Init* com o *Upstart* não altera significativamente os resultados.

```
./configure --prefix=/opt/upstart --
➥sysconfdir=/etc --enable-compat
```

Nesse cenário, será preciso modificar os scripts deixados em `/etc/event.d`. Para isso, simplesmente acrescente a seguinte linha após a linha `script` em `rc-default` e `rcS-sulogin`:

```
export PATH=/opt/upstart/sbin:$PATH
```

Como o diretório do Upstart se localiza no início do caminho de busca, os scripts utilizarão o novo comando *telinit*.

O sistema ainda será inicializado, por padrão, pelo Sys V Init, mas, após a inicialização, é possível mandar o kernel usar o sistema alternativo. A seguinte linha de comando do kernel fará o truque:

```
init=/opt/upstart/sbin/init
```

Para testes simples, talvez seja melhor digitar os parâmetros no *prompt* do carregador de inicialização, mas é possível adicionar um menu de configuração do carregador, caso deseje (**exemplo 4**, **linha 5**).

Análise

O *Bootchart* [4] oferece aos administradores um ótimo método para comparar os dois sistemas de inicialização. A ferramenta registra a carga da CPU e do disco rígido ao longo do processo de inicialização, convertendo os resultados em um belo gráfico. Para permitir que isso aconteça, é necessário instalar o pacote do Bootchart e acrescentar uma entrada à linha de comando do kernel. O *bootchartd* roda como um processo inicial, iniciando o próprio processo *init*.

O **exemplo 5** mostra a entrada necessária no Grub. Se você estiver rodando o Upstart junto com o Init legado, informe esse fato ao Bootchart com o seguinte comando acrescentado à linha do kernel:

```
bootchart_init=/opt/upstart/sbin/init
```

Depois disso, o Bootchart passará a registrar todos os dados interessantes de processos a cada 0,2 segundos, guardando as informações em `/var/log/bootchart.tgz` uma vez que o processo de inicialização tenha sido completado. O comando `bootchart -f png` gera um gráfico *PNG* a partir dos dados, com *SVG* e *EPS* como opções adicionais.

Se compararmos o gráfico de inicialização do Sys V, na **figura 4**, com o do Upstart, na **figura 5**, os resultados poderão ser enganosos. Em nossa máquina de testes, o Bootchart relatou que o Sys V Init levou 33 segundos, enquanto o Upstart precisou de apenas 23 segundos para iniciar o sistema por completo.

Ao conferirmos esses resultados com um cronômetro, vimos que os ganhos efetivos foram de apenas dois segundos. O Bootchart pára o relógio assim que o KDM ou outro gerenciador de login é iniciado. O fato de o Upstart iniciar as tarefas em paralelo significa que esse passo simplesmente começa mais cedo, antes de todos os outros processos críticos de inicialização terminarem.

Seria injusto com o projeto Upstart ignorar sua promesa com base nos resultados atuais dos testes. É importante lembrar que o sucessor do init deverá ser executado em modo de compatibilidade até atingir um nível maior de progresso. Pode-se esperar grandes ganhos de velocidade assim que os scripts de inicialização individuais tenham sido adaptados para suportar o novo sistema. Portanto, o Ubuntu não tem a expectativa de grandes acelerações na inicialização até que o Edgy seja sucedido pelo *Feisty Fawn*.

Conclusão

A época da inicialização em poucos segundos passou e as "cirurgias" executadas pelas distribuições para melhorar o desempenho provavelmente não solucionarão isso. O futuro parece bom para novos projetos como o Upstart.

Mesmo que o Upstart não proporcione a mesma sensação de "ligue e use" que se tem em um console de jogos, ele certamente necessita de menos paciência do que o sistema legado. Obviamente é necessário um bom grau de conhecimentos administrativos para migrar uma máquina Linux em funcionamento, sem exigir uma reinstalação.

Esperamos que o Upstart tenha como objetivo fazer mais do que simplesmente rejuvenescer o processo de inicialização: os desenvolvedores visam a construir um daemon central de serviços que assumirá as tarefas atualmente atribuídas a ferramentas diversas. Isso inclui executar eventos específicos em momentos determinados, ou seja, substituir também o cron e o at. Até lá, o Upstart ainda terá que demonstrar sua capacidade de organizar o processo de inicialização de uma forma sensata e, neste exato momento, ele caminha a passos largos nessa direção. ■

Mais Informações

[1] Upstart do Ubuntu: http://upstart.ubuntu.com

[2] Upstart no braço Experimental do Debian: http://packages.debian.org/experimental/admin/upstart

[3] Upstart no blog de Scott James Remnant: http://www.netsplit.com/blog/articles/2006/08/26/upstart-in-universe

[4] Bootchart: http://www.bootchart.org/