Bossa – Aspektorientierte Scheduler Entwicklung

Marc Lörner, 30.06.2003

(gerd.loerner@t-online.de)

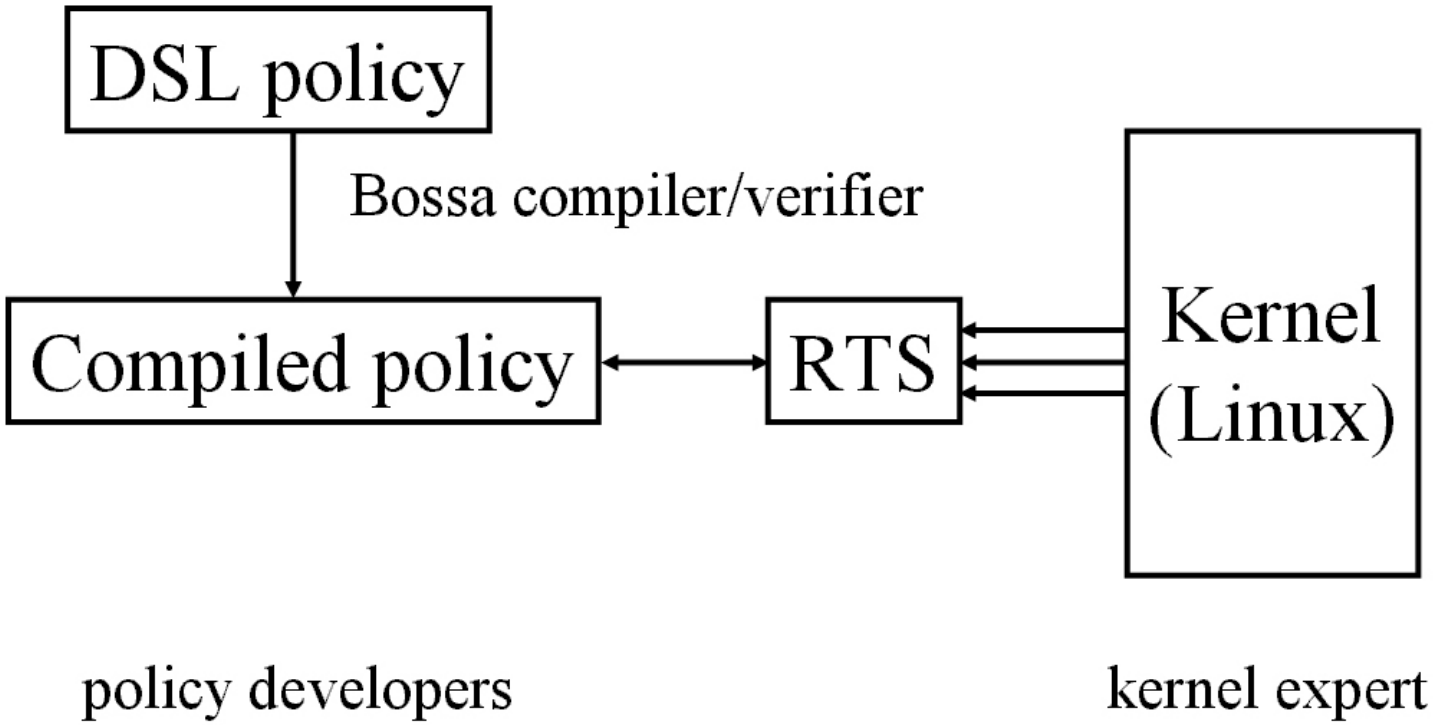
Übersicht

- Motivation
- Arbeitsweise Bossa
- Bossa-DSL
- Scheduler-Hierarchien
- Bossa als Aspekt?
- Performanz und Effizienz
- Zusammenfassung

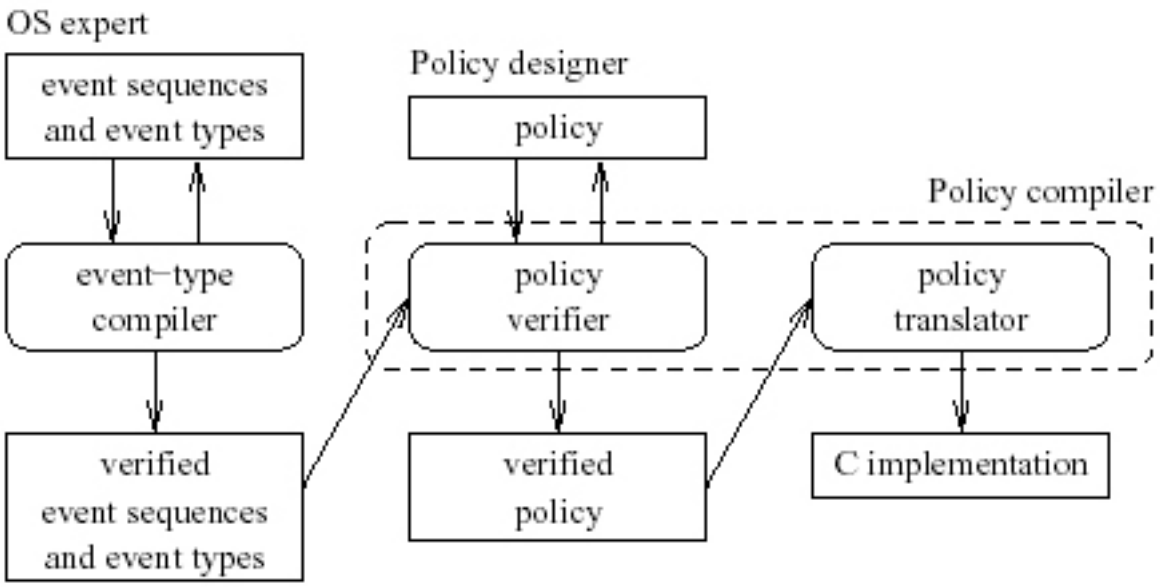
Motivation

- Verteilter Code im Kernel
- Fehlervermeidung bei der Entwicklung
- Effizienzsteigerung
- Einfachere Handhabung, und damit schnellere Entwicklung

Arbeitsweise von Bossa (1)



Arbeitsweise von Bossa (2) (jetzt detailierter)



Die Bossa-DSL

- Deklarationen
- Eventhandler
- Interface-Funktionen

Deklarationen (1)

(aus der Linux 2.2 Scheduling-Strategie)

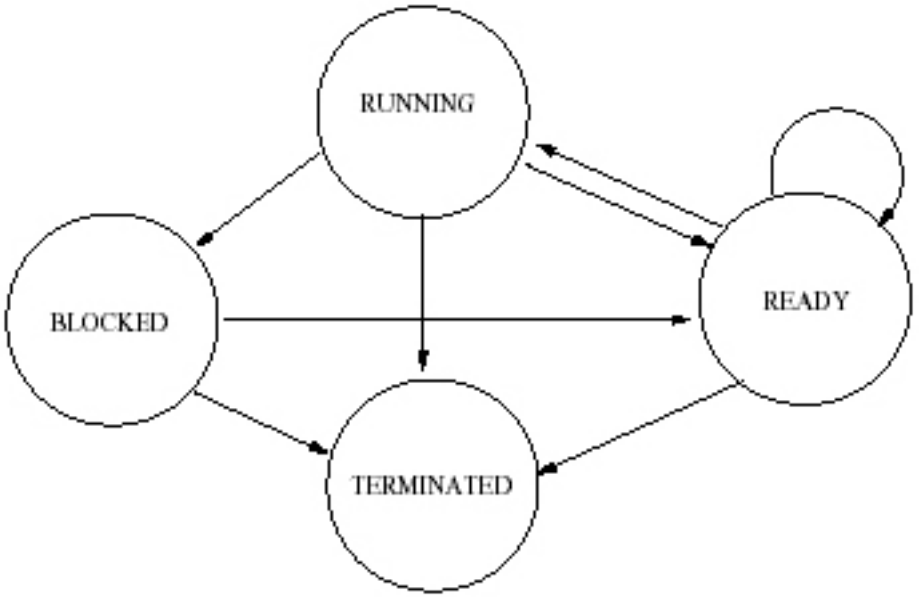
```
Type policy_t =
      enum {SCHED_FIFO, SCHED_RR, SCHED_OTHER};

Process = {
      policy_t                  policy;
      int                       rt_priority;
      time                priority;
      time                ticks
      system struct ctx mm
}
```

Deklarationen (2)

```
States = {
   RUNNING    running     :     process,
                                previous old_running;
   READY      ready       :     sorted queue;
   READY      expired     :     queue;
   BLOCKED    yield       :     process;
   BLOCKED    blocked     :     queue;
   TERMINATED terminated;
}
```

Deklarationen (3)



Deklarationen (4)

```
Ordering_criteria = {
   highest rt_priority, highest ticks,
   highest ((!empty(old_running) &&
             mm == old_running.mm) ? 1:0)
}
```

Eventhandler

(aus der Linux 2.2 Scheduling-Strategie)

```
On block.* {
   e.target => blocked;
}

On unblock.* {
   if(e.target in blocked) {
        e.target => ready;
        if(!empty(running) && e.target > running) {
               running => ready;
        }
   }
}
```

Interface-Funktionen

```
Void
sched_setscheduler(process p, policy_t policy,
                        system struct sched_param param){
   int prio = param.sched_priority;
   if(policy == SCHED_OTHER && prio != 0) {
       error(„non-RT process: invalid priority“);
   } else {
       if(prio < 0 || prio > 99) {
              error(„RT process: invalid priority“);
       }
   }
   p.policy = policy;
   p.rt_priority = prio;
}
```

Scheduler-Hierarchien

- Als Baum organisiert
- Interne Knoten entsprechen virtuellen Schedulern
- Blätter entsprechen Prozess-Schedulern

Probleme bei Hierarchien

- Events durch den ganzen Baum geleitet
- Bossa-Primitiven: - `forwardImmediate`
  - `forwardDeferred`
- Virtuelle Scheduler müssen Schnittstellen für Hinzufügung von neuen Schedulern haben

Beispiel für zusätzl. Schnittstellen

```
Void attach(process p, policy_t policy,
            int rt_priority, time priority) {
  p.policy = policy;
  p.rt_priority = rt_priority;
  p.priority = priority;
  p => ready;
}
```

Beispiel für Events in Hierarchien

```
On block.*{
   e.next => forwardImmediate();
}

On unblock.* {
   if(e.next in blocked) {
        e.next => forwardImmediate();
        if(e.next in ready && !empty(running) &&
           e.next > running) {
                running => ready;
        }
   } else {
        e.next => forwardImmediate();
   }
}
```

Bossa als Aspekt?

- Scheduling-Code besitzt hohen Grad der Verteilung in Betriebssystemen

$\Rightarrow$ Scheduling-Aspekt

- Bossa ist Event-basiertes System

=> Event-based model of AOP (EAOP)

EAOP

- Joinpoints sind Events
- Event „matches“ => Hauptprogramm wird verlassen, Eventcode wird ausgeführt und danach Hauptprogramm fortgesetzt

⇒ Dynamisches Weben

Performanz und Effizienz

Benchmark	parameter	Linux	Bossa	Bossa average/Linux average
lat_pipe		4.221 ± 0.008	4.404 ± 0.057	104%
lat_proc	null dummy	90.22 ± 0.16	93.87 ± 0.22	104%
	sh dynamic	15558 ± 23	14036 ± 26	90%
	sh static	14457 ± 10	12887 ± 17	89%
	simple dynamic	1253.3 ± 6.3	1259.0 ± 3.7	100%
	simple static	195.4 ± 4.3	183.3 ± 1.7	94%
lat_fs	create 0	9463 ± 25	9307 ± 14	98%
lat_ctx	0k 2	1.130 ± 0.015	1.240 ± 0.024	109%
	0k 20	1.916 ± 0.033	2.147 ± 0.193	112%
lat_ctx2	4k 20	2.625 ± 0.201	2.492 ± 0.098	95%
	32k 20	52.141 ± 0.292	52.971 ± 0.199	102%
	64k 20	103.256 ± 0.091	104.510 ± 0.062	101%

Zusammenfassung

- Bossa erleichtert Scheduler-Entwicklung
- Bossa fast genauso effizient wie handgeschriebener Scheduler