Linux Kernel 3.19 Modifications

**/kernel\_source\_code/arch/x86/kconfig file:**

menu "Background Scheduler"

config SCHED\_BG\_POLICY

bool "Background Cpu Scheduler"

help

New scheduler for linux.

Endmenu

**pathname/include/uapi/linux/sched.h:**

#ifdef CONFIG\_SCHED\_BG\_POLICY

#define SCHED\_BG 7

#endif

**pathname/include/linux/sched.h:**

**In task\_struct definition:**

#ifdef CONFIG\_SCHED\_BG\_POLICY

unsigned int bg\_id;

#endif

struct sched\_bg\_entity bg;

**In sched\_param definition:**

#ifdef CONFIG\_SCHED\_BG\_POLICY

unsigned int bg\_id;

#endif

extern void

print\_bg\_rq(struct seq\_file \*m, int cpu, struct bg\_rq \*bg\_rq);

struct sched\_bg\_entity{

EL CONTENIDO ES UNA COPIA DEL SCHED\_ENTITY.

}

**pathname/kernel/sched/sched.h:**

extern struct sched\_bg\_entity \*\_\_pick\_bg\_first\_entity(struct bg\_rq \*bg\_rq);

extern struct sched\_bg\_entity \*\_\_pick\_bg\_last\_entity(struct bg\_rq \*bg\_rq);

extern void print\_bg\_stats(struct seq\_file \*m, int cpu);

extern void init\_bg\_rq(struct bg\_rq \*bg\_rq);

extern void \_\_refill\_bg\_bandwidth\_runtime(struct bg\_bandwidth \*bg\_b);

extern void \_\_start\_bg\_bandwidth(struct bg\_bandwidth \*bg\_b, bool force);

extern void unthrottle\_bg\_rq(struct bg\_rq \*bg\_rq);struct bg\_rq

extern void init\_bg\_bandwidth(struct bg\_bandwidth \*bg\_b);

extern void init\_tg\_bg\_entry(struct task\_group \*tg,

struct bg\_rq \*bg\_rq,

struct sched\_bg\_entity \*se,

int cpu,

struct sched\_bg\_entity \*parent);

extern void free\_bg\_sched\_group(struct task\_group \*tg);

extern int alloc\_bg\_sched\_group(struct task\_group \*tg, struct task\_group \*parent);

extern void unregister\_bg\_sched\_group(struct task\_group \*tg, int cpu);

{

EL CONTENIDO ES UNA COPIA DEL CFS\_RQ

};

Struct bg\_bandwidth: COPIA DE cfs\_bandwidth.

**Se agregó a la definicion de struct\_rq:**

struct bg\_rq bg\_rq;

struct list\_head leaf\_bg\_rq\_list;

struct list\_head bg\_tasks;

extern const struct sched\_class bg\_sched\_class;

**in task group:**

struct sched\_bg\_entity \*\*bg;

struct bg\_rq \*\*bg\_rq;

struct bg\_bandwidth bg\_bandwidth;

NOTA: Considero que el bg\_task debe ir declarado como sched\_bg\_entity en el archivo include/Linux/sched.h como lo hacen el resto de schedulers!

NOTA2: El contenido del task es la copia del contenido del sched\_entity definido en el archivo de la nota pasada.

**pathname/kernel/sched/core.c:**

**in** \_\_init init\_sched():

init\_bg\_rq(&rq->bg\_rq);

init\_sched\_bg\_class();

DE PRONTO TOCA HACER MÁS CAMBIOS EN ESTE MÉTODO.

**in fair\_policy(int policy):**

return policy == SCHED\_NORMAL || policy == SCHED\_BATCH || policy == SCHED\_BG;

**In \_\_setscheduler(…):**

else if (p->policy == SCHED\_BG)

p->sched\_class = &bg\_sched\_class;

**NOTAS DE IMPLEMENTACION DE LA CLASE BG:**

**Métodos que de pronto se deben cambiar:**

list\_del\_rcu

nr\_numa\_running (Se deberian crear copias de todas estas variables?)

NOTA:

TODAVIA HAY METODOS EN EL FAIR CLASS QUE SE SALTA A IDLE PARA OPTIMIZAR

FALTA PRINT\_BG\_RQ