in kernel/smp.c

```c
/* Called by boot processor to activate the rest. */
void __init smp_init(void)
{
        unsigned int cpu;

        idle_threads_init();                          ①

        /* FIXME: This should be done in userspace --RR */
        for_each_present_cpu(cpu) {
                if (num_online_cpus() >= setup_max_cpus)
                        break;
                if (!cpu_online(cpu))
                        cpu_up(cpu);                          ②
        }

        /* Any cleanup work */
        smp_announce();
        smp_cpus_done(setup_max_cpus);
}
```

①

Linux kernel为每个core维护着一个idle process，即如果没有任何task可运行时，就运行它。

所以这里要为即将满血复活的secondary core生成对应的idle process。

in kernel/smpboot.c

```c
/**
 * idle_threads_init - Initialize idle threads for all cpus
 */
void __init idle_threads_init(void)
{
        unsigned int cpu, boot_cpu;

        boot_cpu = smp_processor_id();

        for_each_possible_cpu(cpu) {
                if (cpu != boot_cpu)                            (A)
                        idle_init(cpu);                         (B)
        }
}
```

(A)

排除boot cpu

(B)

```
/**
 * idle_init - Initialize the idle thread for a cpu
 * @cpu:        The cpu for which the idle thread should be initialized
 *
 * Creates the thread if it does not exist.
 */
static inline void idle_init(unsigned int cpu)
{
        struct task_struct *tsk = per_cpu(idle_threads, cpu);
                (C)

        if (!tsk) {
                tsk = fork_idle(cpu);
                                                        (D)
                if (IS_ERR(tsk))
                        pr_err("SMP: fork_idle() failed for CPU %u\n", cpu);
                else
                        per_cpu(idle_threads, cpu) = tsk;
        }
}

struct task_struct *fork_idle(int cpu)
{
        struct task_struct *task;
        task = copy_process(CLONE_VM, 0, 0, NULL, &init_struct_pid, 0);        (
E)
        if (!IS_ERR(task)) {
                init_idle_pids(task->pids);
                init_idle(task, cpu);
        }

        return task;
}
```

(C)

idle process的task_struct *是个per_cpu variable。

(D)

create idle process

(E)

idle process完全运行在kernel mode，所以它与boot cpu的idle process是share virtual memory的。

当secondary core初始化好以后，运行的第一个process code就是该core对应的idle process。

②

in kernel/cpu.c

```
int cpu_up(unsigned int cpu)
{
        int err = 0;

        if (!cpu_possible(cpu)) {
                pr_err("can't online cpu %d because it is not configured as may-hotadd at boot time\n",
                       cpu);
#if defined(CONFIG_IA64)
                pr_err("please check additional_cpus= boot parameter\n");
#endif
                return -EINVAL;
        }

        err = try_online_node(cpu_to_node(cpu));
        if (err)
                return err;

        cpu_maps_update_begin();

        if (cpu_hotplug_disabled) {
                err = -EBUSY;
                goto out;
        }

        err = _cpu_up(cpu, 0);

out:
        cpu_maps_update_done();
        return err;
}

static int _cpu_up(unsigned int cpu, int tasks_frozen)
{
        int ret, nr_calls = 0;
        void *hcpu = (void *)(long)cpu;
        unsigned long mod = tasks_frozen ? CPU_TASKS_FROZEN : 0;
        struct task_struct *idle;

        cpu_hotplug_begin();

        if (cpu_online(cpu) || !cpu_present(cpu)) {
                                (A)
                ret = -EINVAL;
                goto out;
        }

        idle = idle_thread_get(cpu);
                                        (B)
        if (IS_ERR(idle)) {
                ret = PTR_ERR(idle);
                goto out;
        }

        ret = smpboot_create_threads(cpu);
                                (C)
        if (ret)
                goto out;

        ret = __cpu_notify(CPU_UP_PREPARE | mod, hcpu, -1, &nr_calls);          (
D)
        if (ret) {
                nr_calls--;
                pr_warn("%s: attempt to bring up CPU %u failed\n",
                        __func__, cpu);
                goto out_notify;
        }

        /* Arch-specific enabling code. */
        ret = __cpu_up(cpu, idle);
                                        (E)
        if (ret != 0)
                goto out_notify;
        BUG_ON(!cpu_online(cpu));

        /* Wake the per cpu threads */
        smpboot_unpark_threads(cpu);
                                        (F)

        /* Now call notifier in preparation. */
        cpu_notify(CPU_ONLINE | mod, hcpu);
                                (G)

out_notify:
        if (ret != 0)
                __cpu_notify(CPU_UP_CANCELED | mod, hcpu, nr_calls, NULL);
out:
        cpu_hotplug_done();

        return ret;
}
```

(A)

如果该core已经活了或不存在，自然skip。

(B)

取得对应core的idle process指针(task_struct *),也就是前一步创建的idle process。

(C)

每个core还有hotplug thread，没研究。

(D)

observer pattern的应用。发出CPU_UP_PREPARE notification。

(E)

这是boot secondary core的核心,见后分析。如果该function成功返回，那么该core就online了。

(F)

同样是cpu hotplug相关操作

(G)

observer pattern的应用。发出CPU_ONLINE notification。