HVM Scalability

- **Shared Device Model Bottleneck**
  - Device emulation for all hvm guests performed in dom0 userspace
  - hvm guest isolation is weak due to shared device model

- **Distribute the Qemu Device Model into “Stub Domains”**
  - Give each HVM Domain a dedicated instance of qemu-dm
  - [http://xenbits.xensource.com/ext/stub-domain.hg](http://xenbits.xensource.com/ext/stub-domain.hg)
  - Ryan Harper

- **Improve Emulated Devices**
  - Ryan Harper

- **Implement Scheduling Domains**
  - **Nemesis Exokernel**
    - Allow grouping of virtual machines into *scheduling domains* (sdom)
    - Each sdom manages its own time slice and distributes time among *activation domains* (adom)
New domctrl sub-operation to manage sdom-adom associations

```c
    case XEN_DOMCTL_scheduler_op:
        {
            struct domain *d;

            if (op->u.scheduler_op.cmd == XEN_DOMCTL_SCHEDOP_sdom) {
                /* handle sdom & adom operations here */
                ret = do_sdom_op(&op->u.scheduler_op.u.sdom);
                break;
            }

            ...

            adom = get_domain_by_id(domctl->adom);
            if (adom) {
                if ( ! (adom->sched_priv->flags & SDOM_ACTIVE ) ) {
                    spin_lock(&sdom->sched_priv->pdom_write_lock);
                    add_adom_to_sdom(adom->sched_priv, sdom->sched_priv);
                    spin_unlock(&sdom->sched_priv->pdom_write_lock);
                    ret = 0;
                } else {
                    domctl->reason = SDOM_err_already_sdom;
                    put_domain(adom);
                }
            }
            put_domain(sdom);
```
Additional domain scheduling attributes

/* flags for nemesis-style scheduling domains */
#define SDOM_INACTIVE  0L
#define SDOM_ACTIVE     1
#define SDOM_IS_ADOM    (SDOM_ACTIVE << 1)

struct sched_priv_domain
{
    struct list_head adom_list;
    spinlock_t pdom_write_lock;
    struct sched_priv_domain *sdom;
    uint32_t flags;
    struct domain *domain;
    union
    {
        struct csched_dom {
            struct list_head active_vcpu;
            struct list_head active_sdom_elem;
            uint16_t active_vcpu_count;
            uint16_t weight;
            uint16_t cap;
        } credit_dom;
        struct sedom_info {
            } sedom;
    };
};
Credit Scheduler Implementation

/* if the running vcpu is owned by an activation domain, redirect the */
/* accounting and scheduling functions to the parent (sdm) */
static inline struct csched_dom * SCHEDULING_DOMAIN(struct csched_dom *dom)
{
    struct sched_priv_domain *spd = SCHED_PRIV_DOM(dom);
    if (spd->flags & SDOM_IS_ADOM)
        return &spd->sdom->credit_dom;
    else
        return dom;
}

...

static inline void
__csched_vcpu_acct_start(struct csched_vcpu *svc)
{
    unsigned long flags;
    struct csched_dom * const sdom = SCHEDULING_DOMAIN(svc->sdom);

    spin_lock_irqsave(&csched_priv.lock, flags);
    ...

The credit scheduler is driven by vcpu accounting, so redirecting
adom accounting to the sdom causes them to share time slices