在 kernel/irq/manage.c:

```
#ifndef CONFIG_AUTO_IRQ_AFFINITY
/*
 * Generic version of the affinity autoselector.
 */
int irq_setup_affinity(struct irq_desc *desc)
{
	struct cpumask *set = irq_default_affinity;
	int ret, node = irq_desc_get_node(desc);
	static DEFINE_RAW_SPINLOCK(mask_lock);
	static struct cpumask mask;

	/* Excludes PER_CPU and NO_BALANCE interrupts */
	if (!__irq_can_set_affinity(desc))
		return 0;

	raw_spin_lock(&mask_lock);
	/*
	 * Preserve the managed affinity setting and a userspace affinity
	 * setup, but make sure that one of the targets is online.
	 */
	if (irqd_affinity_is_managed(&desc->irq_data) ||
	    irqd_has_set(&desc->irq_data, IRQD_AFFINITY_SET)) {
		if (cpumask_intersects(desc->irq_common_data.affinity,
				       cpu_online_mask))
			set = desc->irq_common_data.affinity;
		else
			irqd_clear(&desc->irq_data, IRQD_AFFINITY_SET);
	}

	cpumask_and(&mask, cpu_online_mask, set);
	if (cpumask_empty(&mask))
		cpumask_copy(&mask, cpu_online_mask);

	if (node != NUMA_NO_NODE) {
		const struct cpumask *nodemask = cpumask_of_node(node);

		/* make sure at least one of the cpus in nodemask is online */
		if (cpumask_intersects(&mask, nodemask))
			cpumask_and(&mask, &mask, nodemask);
	}
	ret = irq_do_set_affinity(&desc->irq_data, &mask, false);
	raw_spin_unlock(&mask_lock);
	return ret;
}
#else
/* Wrapper for ALPHA specific affinity selector magic */
int irq_setup_affinity(struct irq_desc *desc)
{
	return irq_select_affinity(irq_desc_get_irq(desc));
}
#endif
```

这里顺便提下 CONFIG_AUTO_IRQ_AFFINITY

自动 IRQ 亲和性，在内核配置中可以打开自动 IRQ 亲和性开关（CONFIG_AUTO_IRQ_AFFINITY）使能此功能。可自动平衡中断 irq 在各个处理器上，提升系统的性能。

irq_do_set_affinity 打开：

```
int irq_do_set_affinity(struct irq_data *data, const struct cpumask *mask,
			bool force)
{
	struct irq_desc *desc = irq_data_to_desc(data);
	struct irq_chip *chip = irq_data_get_irq_chip(data);
	int ret;

	if (!chip || !chip->irq_set_affinity)
		return -EINVAL;

	ret = chip->irq_set_affinity(data, mask, force);
	switch (ret) {
	case IRQ_SET_MASK_OK:
	case IRQ_SET_MASK_OK_DONE:
		cpumask_copy(desc->irq_common_data.affinity, mask);
		/* fall through */
	case IRQ_SET_MASK_OK_NOCOPY:
		irq_validate_effective_affinity(data);
		irq_set_thread_affinity(desc);
		ret = 0;
	}

	return ret;
}
```

这里 irq_chip 的 irq_set_affinity 就是 GIC 注册的回调函数了：

```
static struct irq_chip gic_chip = {
        .name                   = "GICv3",
        .irq_mask               = gic_mask_irq,
        .irq_unmask             = gic_unmask_irq,
        .irq_eoi        = gic_eoi_irq,
        .irq_set_type           = gic_set_type,
        .irq_set_affinity       = gic_set_affinity,
        .irq_get_irqchip_state  = gic_irq_get_irqchip_state,
        .irq_set_irqchip_state  = gic_irq_set_irqchip_state,
        .irq_nmi_setup          = gic_irq_nmi_setup,
        .irq_nmi_teardown = gic_irq_nmi_teardown,
        .flags                  = IRQCHIP_SET_TYPE_MASKED |
                                  IRQCHIP_SKIP_SET_WAKE |
                                  IRQCHIP_MASK_ON_SUSPEND,
};

static int gic_set_affinity(struct irq_data *d, const struct cpumask *mask_val,
                            bool force)
{
        unsigned int cpu;
        void __iomem *reg;
        int enabled;
        u64 val;

        if (force)
                cpu = cpumask_first(mask_val);
        else
                cpu = cpumask_any_and(mask_val, cpu_online_mask);

        if (cpu >= nr_cpu_ids)
                return -EINVAL;

        if (gic_irq_in_rdist(d))
                return -EINVAL;

        /* If interrupt was enabled, disable it first */
        enabled = gic_peek_irq(d, GICD_ISENABLER);
        if (enabled)
                gic_mask_irq(d);

        reg = gic_dist_base(d) + GICD_IROUTER + (gic_irq(d) * 8);
        val = gic_mpidr_to_affinity(cpu_logical_map(cpu));

        gic_write_irouter(val, reg);

        /*
         * If the interrupt was enabled, enabled it again. Otherwise,
         * just wait for the distributor to have digested our changes.
         */
        if (enabled)
                gic_unmask_irq(d);
        else
                gic_dist_wait_for_rwp();

        irq_data_update_effective_affinity(d, cpumask_of(cpu));

        return IRQ_SET_MASK_OK_DONE;
}
```

可以看到具体设置亲和力的，是 GIC 内部的中断路由相关的函数。

在/proc 下面，可以查看相应的亲和力的信息：

$ cat /proc/interrupts

CPU0 CPU1

19: 477 1058095 IO-APIC 19-fasteoi ens33 (中断号 irq 19)

$ cat /proc/irq/19/smp_affinity

00000000,00000000,00000000,00000002(2 是十六进制，表示网卡 ens33 的 19 号中断被分配在了 CPU1 上)

$ cat /proc/irq/19/smp_affinity_list

1

$ cat /proc/irq/19/affinity_hint

00000000,00000000,00000000,00000000

$ cat /proc/irq/default_smp_affinity

ffffffff,ffffffff,ffffffff,ffffffff