



2008 Linux 开发者论坛 – 中国

## 为Linux引入技术

*Or: Introducing your technology Into Linux will require introducing a lot of Linux into your technology!!!*

保罗 E. 麦肯尼, 杰出工程师  
IBM Linux 技术中心

# 25周年: 1983年五月至今



# 概览

- 获得**RCU**技术更多资料
- **Linux** 内核中**RCU** 的使用
- **Linux** 如何改变**RCU**
- 经验总结

# 哪里可以获取更多的RCU资料？

# 哪里可以获取更多的RCU资料？

- 在极端要求读的性能、可测量性和决定论的时候，**RCU** 可以被看作读写锁的替代技术
- **RCU**的更多信息请参考：
  - Linux 每周新闻 “真实的 RCU 是什么？” 系列
    - ▶ RCU基本概念？
      - <http://lwn.net/Articles/262464/>
    - ▶ RCU是什么？第二部分：使用
      - <http://lwn.net/Articles/263130/>
    - ▶ RCU 第三部分: RCU API (包括附说明的资料目录)
      - <http://lwn.net/Articles/264090/>
  - Paul McKenney 的 RCU 网址
    - ▶ <http://www.rdrop.com/users/paulmck/RCU/>

# RCU性能问题出现的原因

## 4-CPU 1.8GHz AMD Opteron 844 system

RCU 读取器

Operation	Cost (ns)	Ratio
Clock period	0.6	
Best-case CAS	37.9	63.2
Best-case lock	65.6	109.3
Single cache miss	139.5	232.5
CAS cache miss	306.0	510.0

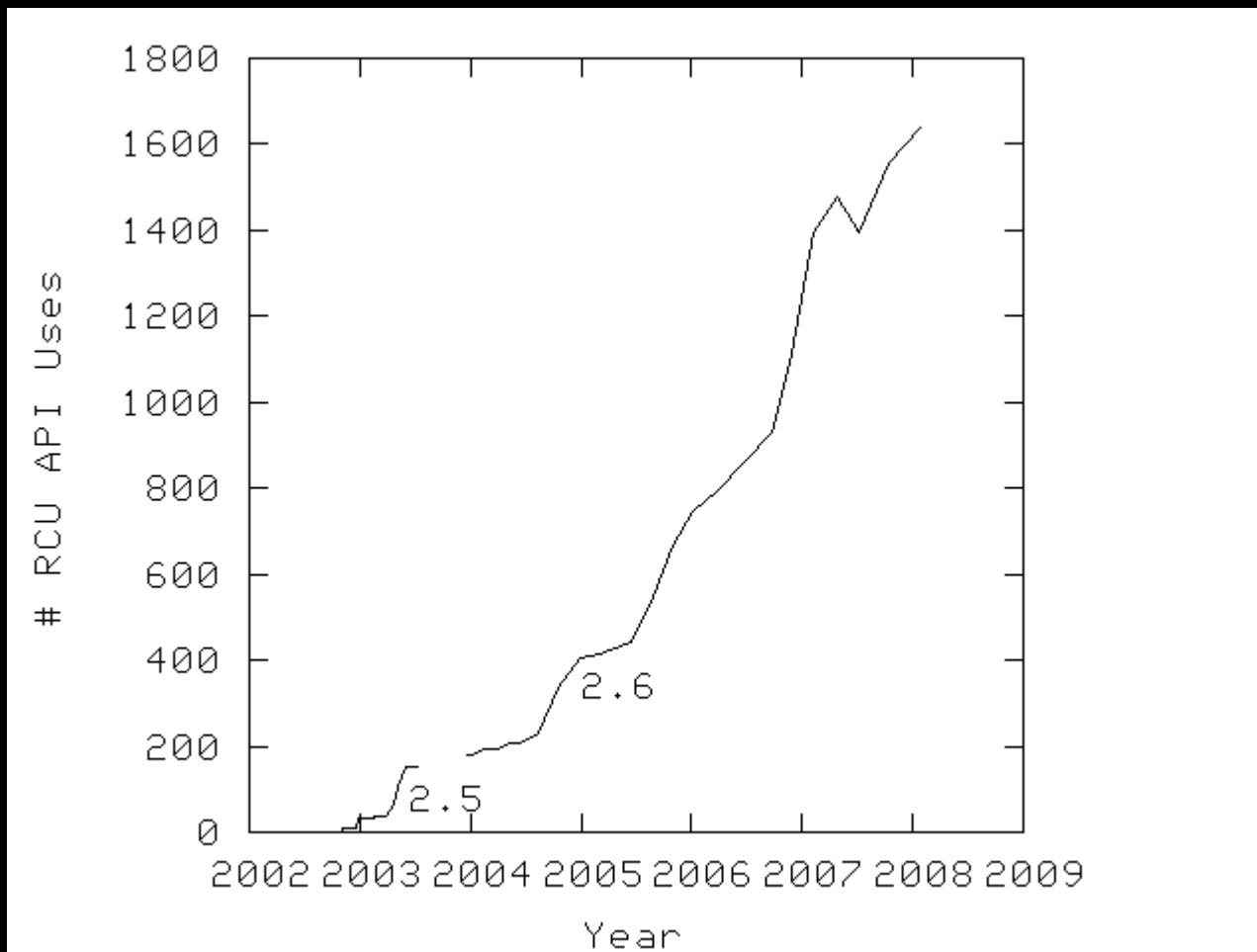
读写器的读取功能可能能高优化 (但是写入器功能不行...)

典型同步机制

RCU 读取器使用低成本指令, 而其他方法使用高成本指令.

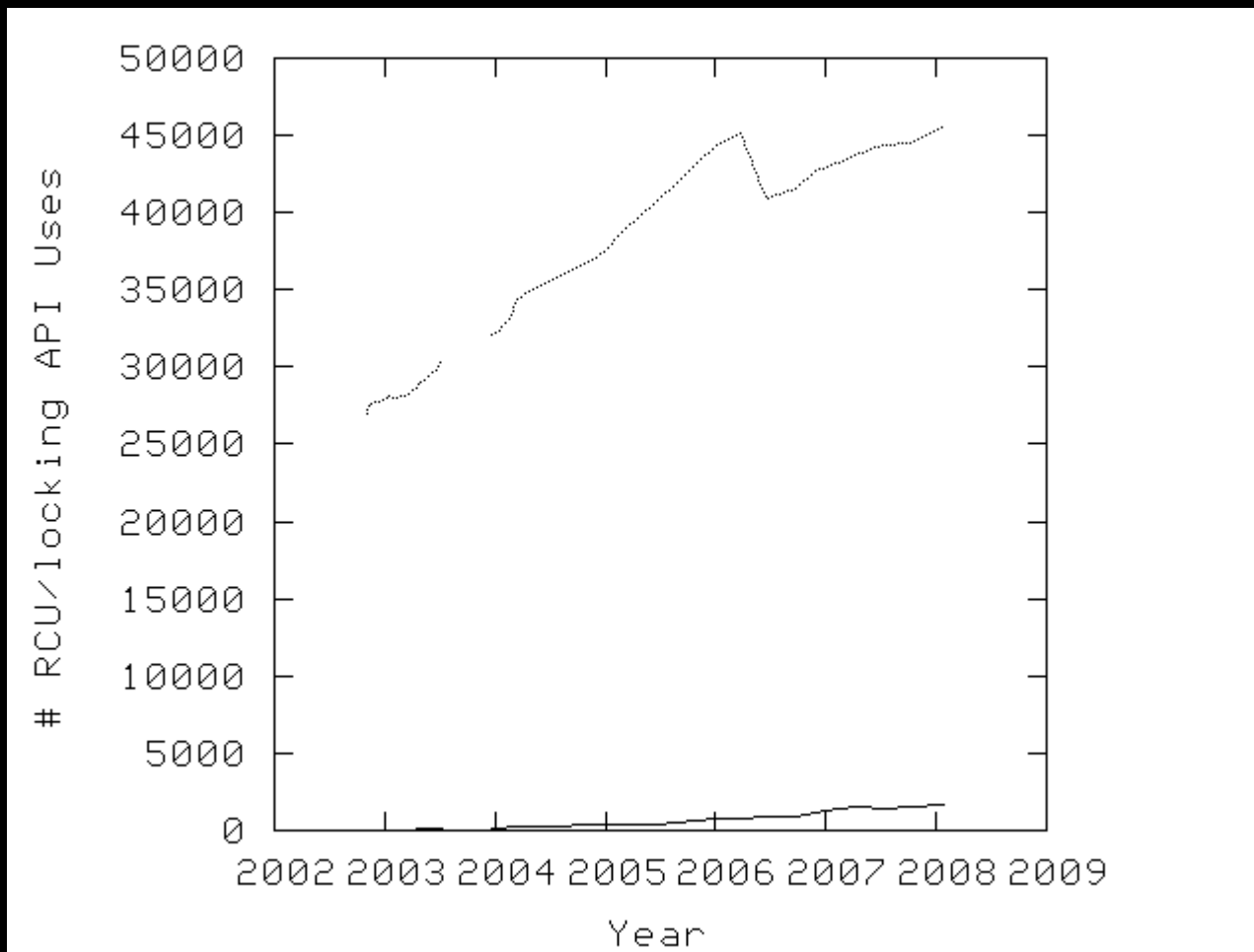
# Linux 内核中RCU 的使用

# Linux 内核中RCU 的使用(2.6.24)

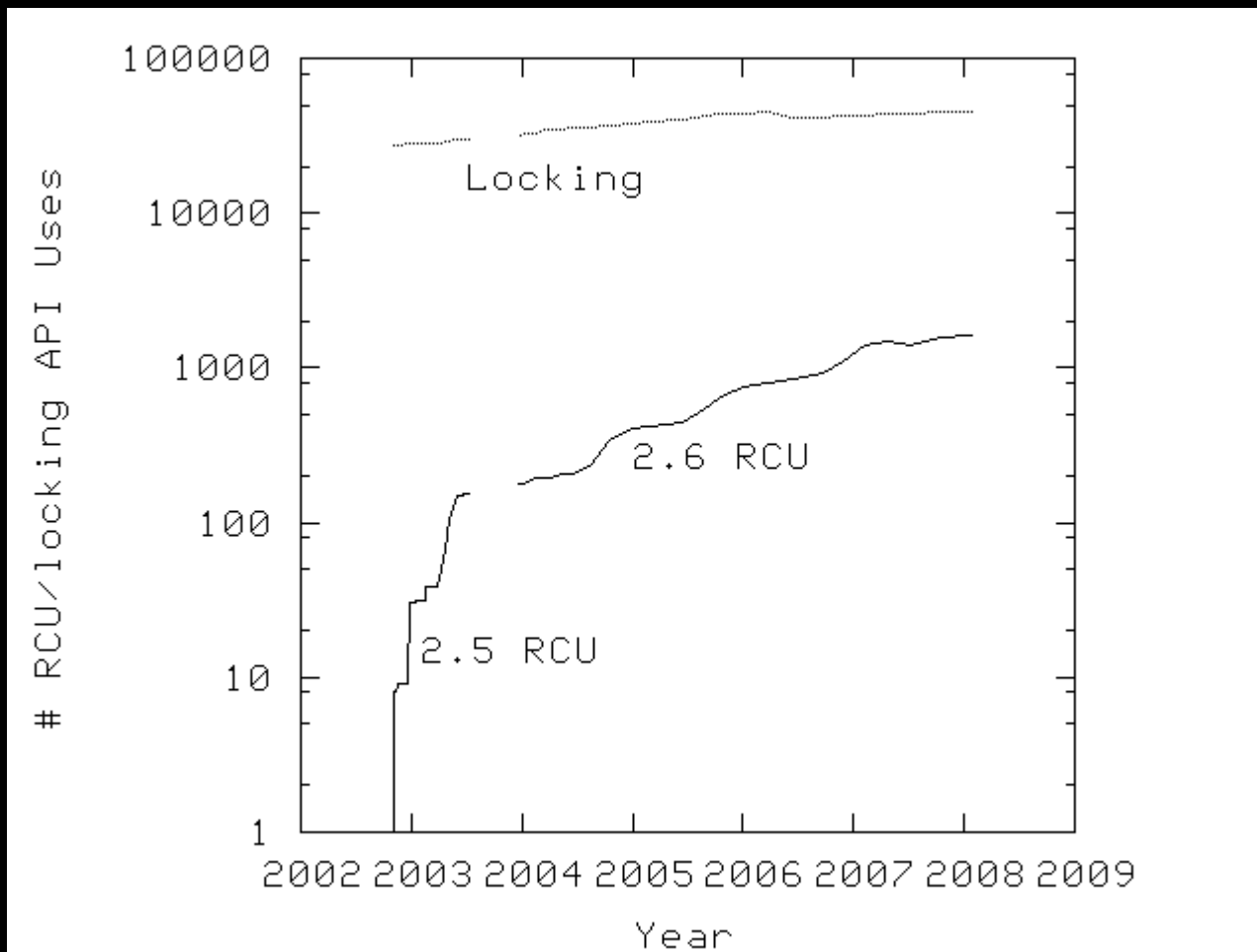


In case there was any doubt, the Linux community *can* handle RCU.

# Linux 内核中RCU 的使用与锁的比较



# Linux 内核中RCU 的使用与锁的比较



# Linux 内核中RCU 的使用与锁的比较

**RCU**技术被成功适当地应用到了Linux内核中

*RCU*如何做到的?

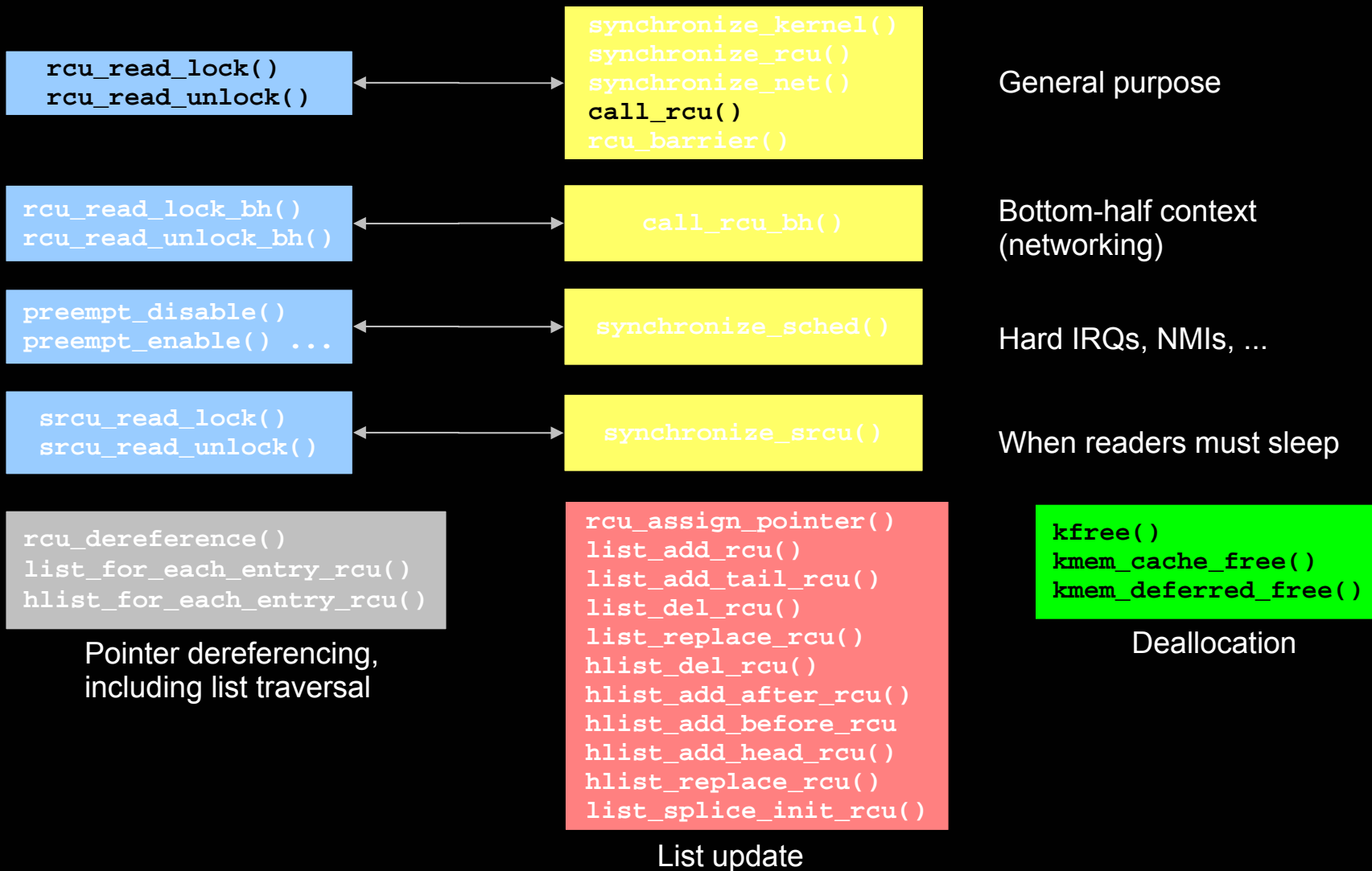
它戏剧性地被Linux系统改变了!!!

# Linux 如何改变RCU

## RCU的已有Linux经验

- 企业系统: 重要并行数据库服务器
  - 10 CPU, 10 GB 内存 (mid-90s 标准)
- 受护网络环境
  - 防火墙, 客户端, 受限使用模式
- 不需实时应答的情形
  - 除非你注重 TPC/A 需求 即90% 事务在2个部分实现
- 内核开发者数目很少 (几十个)
- 为适应 **Linux**做的重大改动
  - 附细节信息和参考供回顾时使用
  - 目的在于给出更改的整体数量级

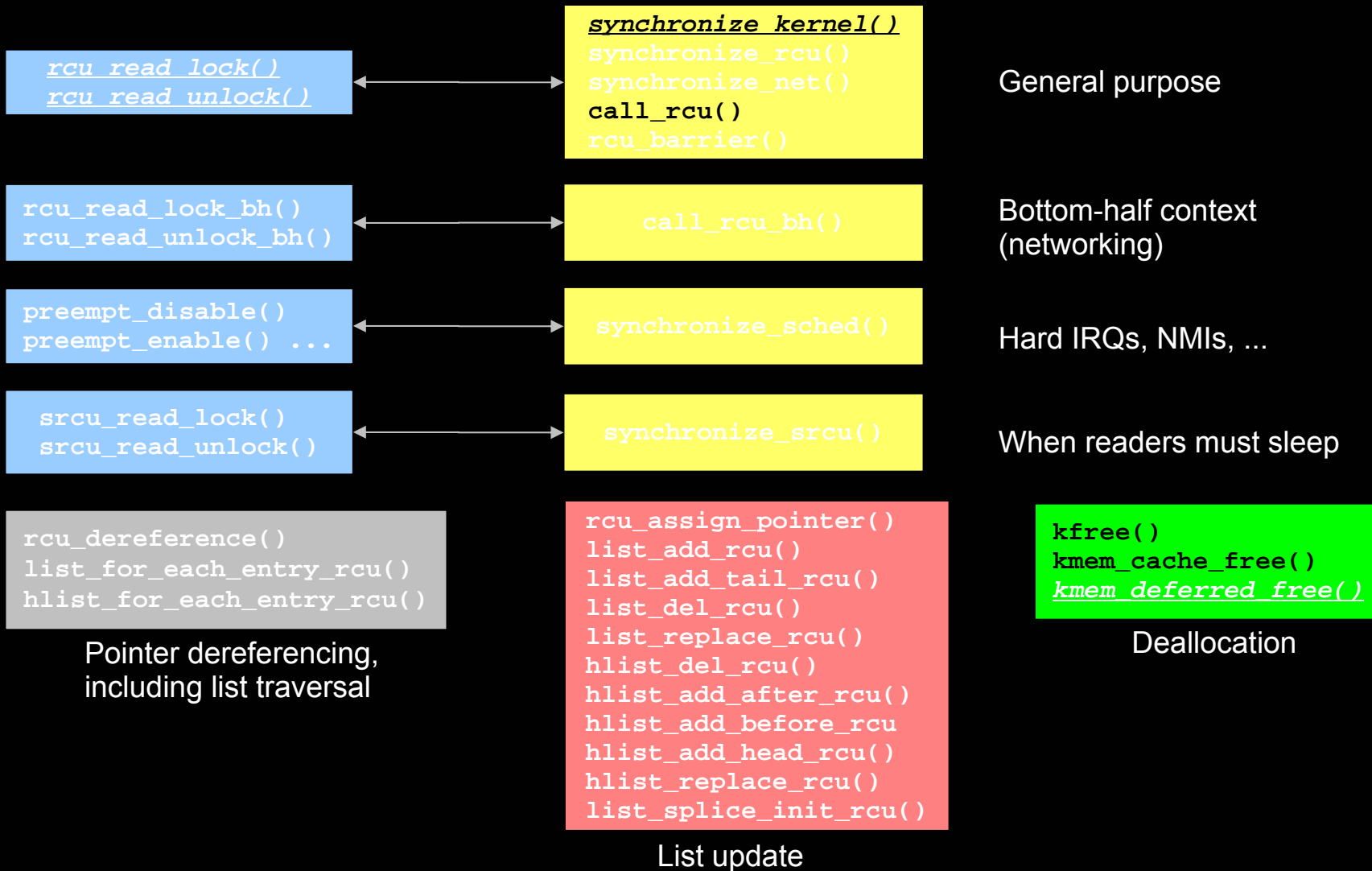
# 基于Linux经验构建RCU API



## 对Linux 内核开发者的价值—简易

- **Painful though it may be at times, this is a good thing**
- **In many cases, complexity is a symptom of lack of understanding**
  - If you know only one way to do something, the odds are against it being the best way!
  - Kudos to Andi Kleen, Rusty Russell, Andrea Arcangeli, and many others for generating alternative implementations
  - And to Dipankar Sarma for doing the implementation and incorporating the plethora of excellent ideas from the Linux community

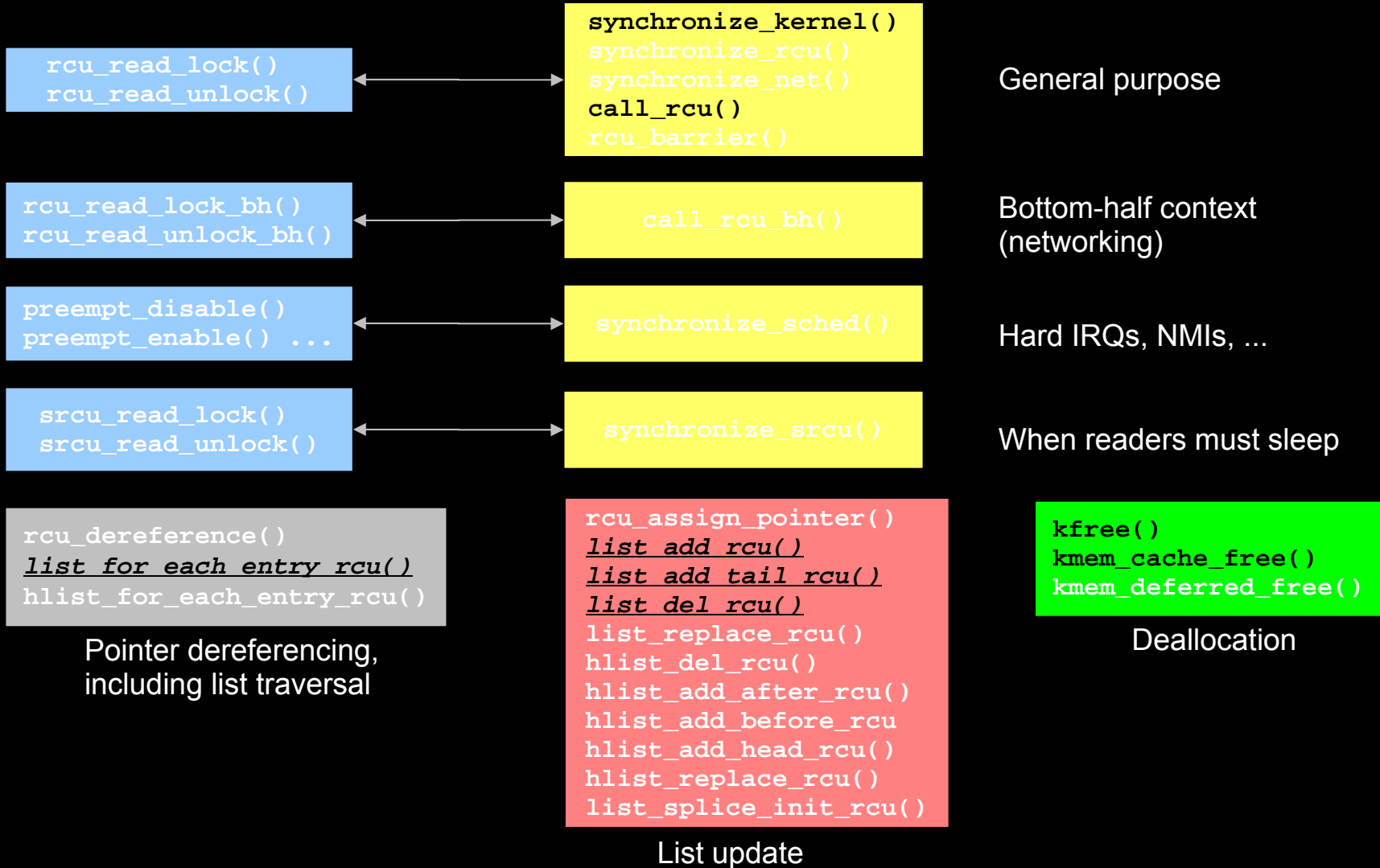
# 简化的Linux RCU API



## 内存栅不受欢迎

- **With good reason**
- **They are hard to understand and easy to get wrong**
- **Many maintainers had a blanket policy:**
  - “Reject any patch containing a memory barrier”
  - This has since been softened to require meaningful comments on memory barriers
- **It is far better to bury any needed memory barriers into a well-designed API**
  - Kudos to Manfred Spraul for suggesting the RCU list API!

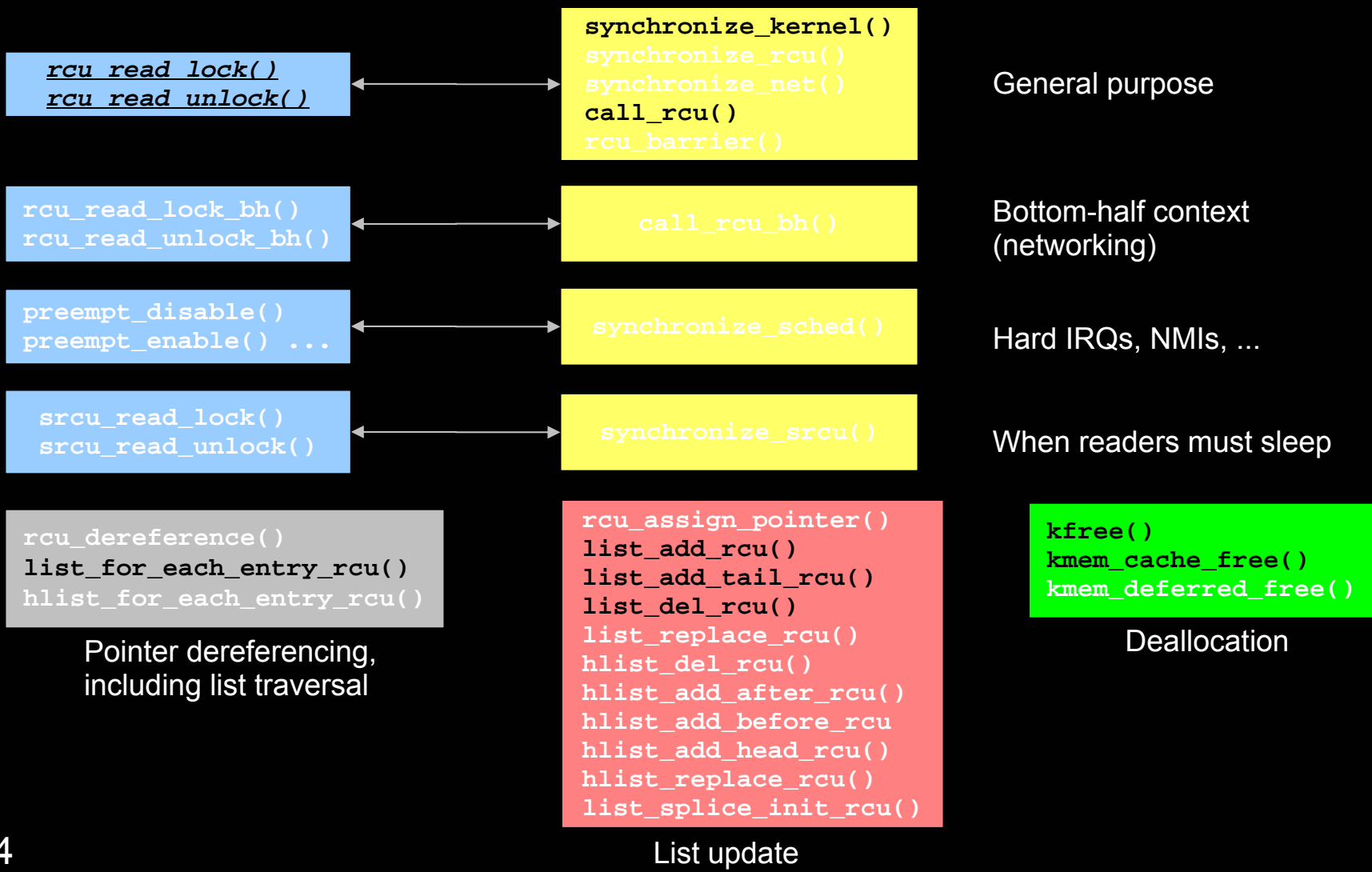
# Linux内核内存栅不受欢迎 (一)



## Linux 内核实时操作系统（一）

- **The CONFIG\_PREEMPT function enters the kernel**
- **The kernel becomes preemptable, invalidating key RCU assumption**
- **Easy fix requires bringing rcu\_read\_lock() and rcu\_read\_unlock() back into the Linux kernel**
  - Where they have been invaluable documentation aids

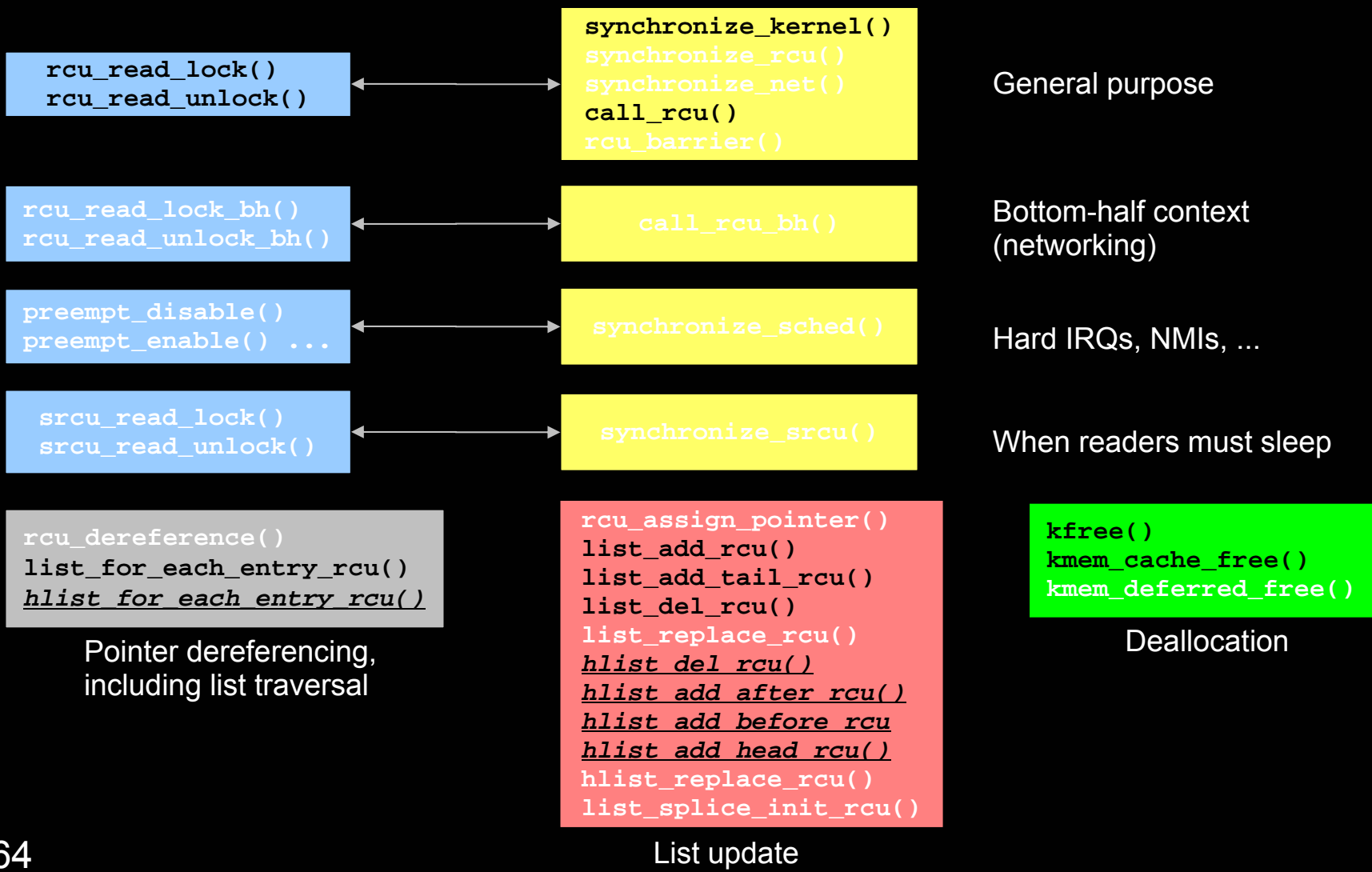
# Linux 内核实时操作系统 (一)



# Linux 内核在小内存系统中运行

- **Linux does circular doubly linked lists**
  - Consumes two pointers per hash bucket
- **Problematic given large hash tables on small-memory machines**
- **Solution: hlist, a linear doubly-linked list**
  - Consumes only one pointer per hash bucket
  - But adds another group of RCU list APIs
  - Implemented by Andi Kleen

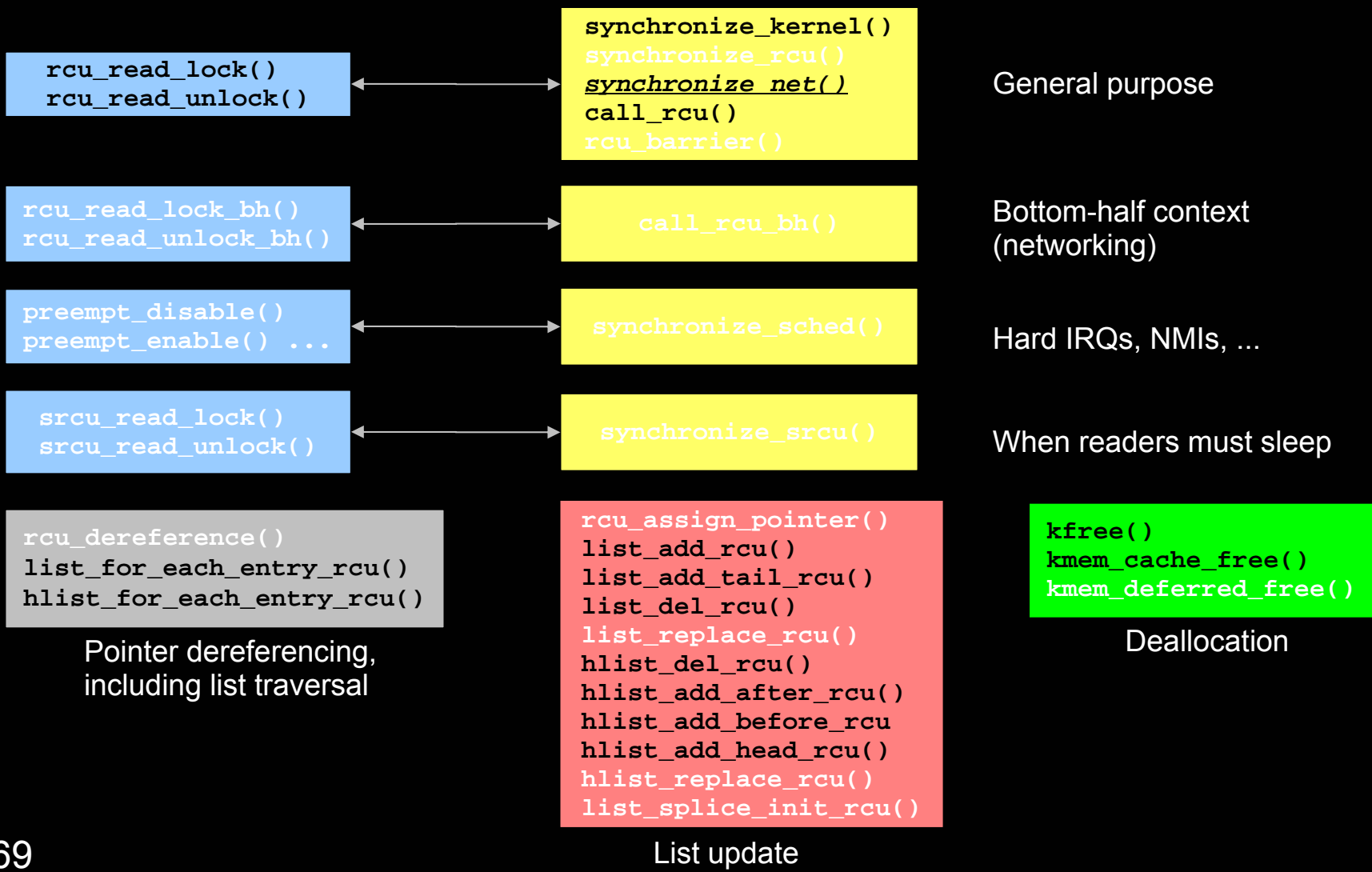
# Linux 内核在小内存系统中运行



## Linux 内核运行繁重网络工作负荷

- **Steve Hemminger converts networking code from brlock to RCU, introducing synchronize\_net() to ease the transition**
  - And synchronize\_net() continues to be a reasonably useful documentation aid

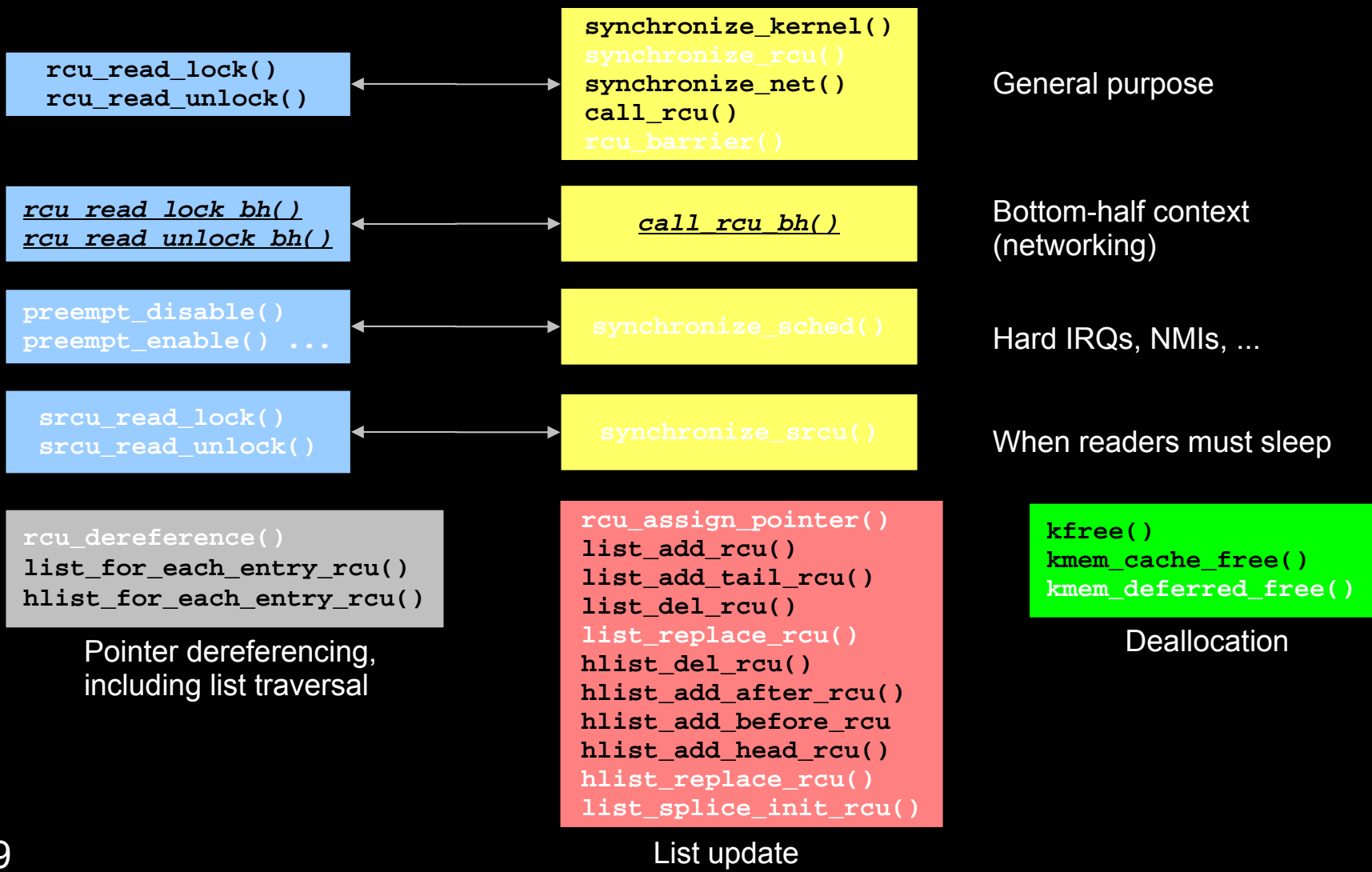
# Linux 内核运行繁重网络工作负荷



## Linux 内核必须抵御网络 DoS 攻击

- **Extremely heavy networking loads from denial-of-service attacks prevent RCU from doing its work**
  - Indefinitely postpones grace periods
- **New `_bh` variant of RCU avoids this problem**
  - Implemented by Dipankar Sarma with Robert Olsson

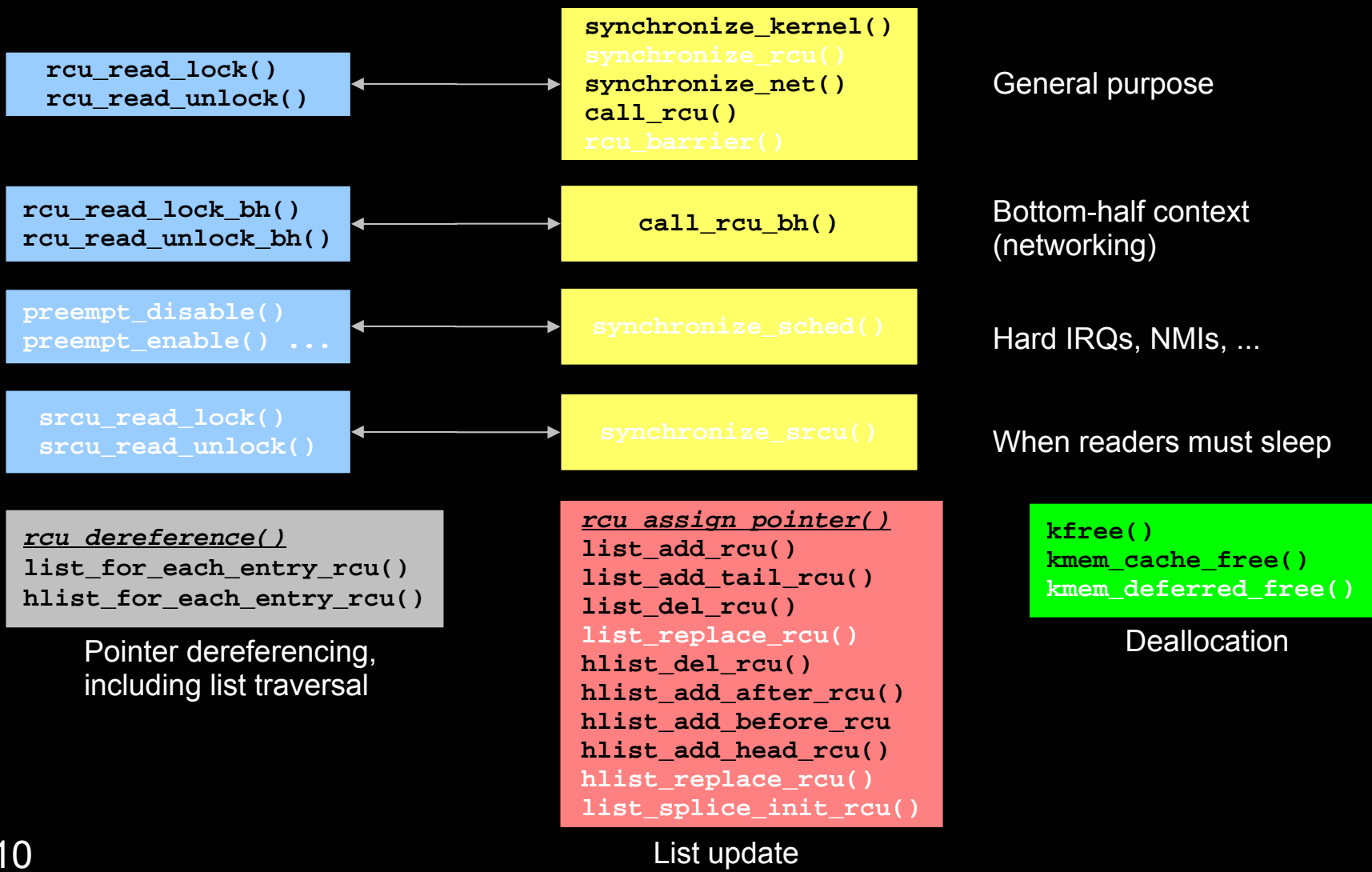
# Linux 内核必须抵御网络 DoS 攻击



## Linux内核内存栅不受欢迎（二）

- **Burying memory barriers in list primitives does not help when applying RCU to non-list data structures**
- **People start applying RCU to trees and the like**
- **Therefore, created primitives to handle this case**

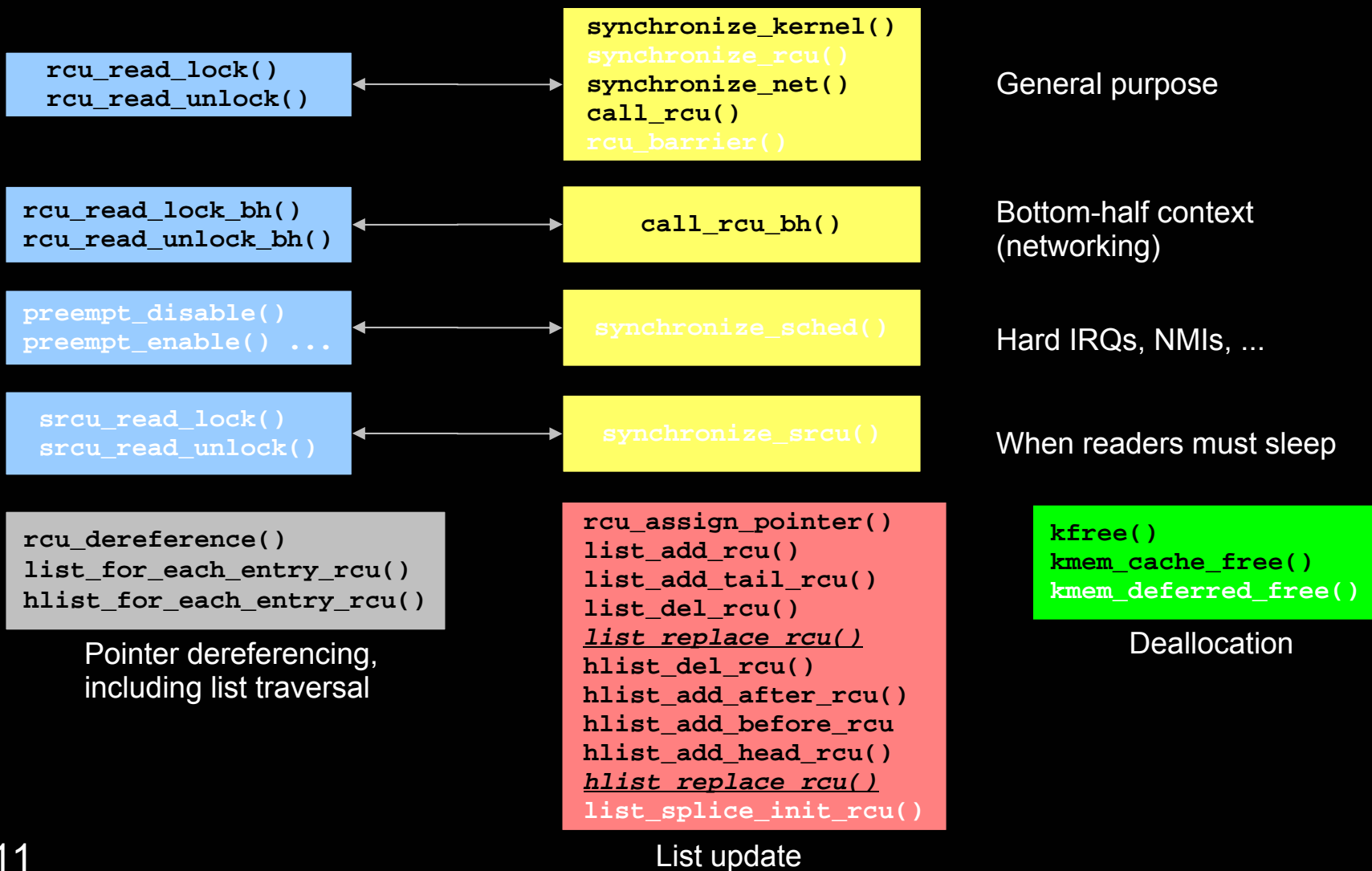
# Linux内核内存栅不受欢迎 (二)



# Linux RCU函数的最终名称和实现

- **“RCU” stands for “read-copy update”**
  - Readers access the data structure with copy-based updates
- **As Murphy would have it, this turned out to be an unusual use case**
- **But the Linux kernel eventually needed it**
  - Kaigai Kohei implements it for SELinux AVC work

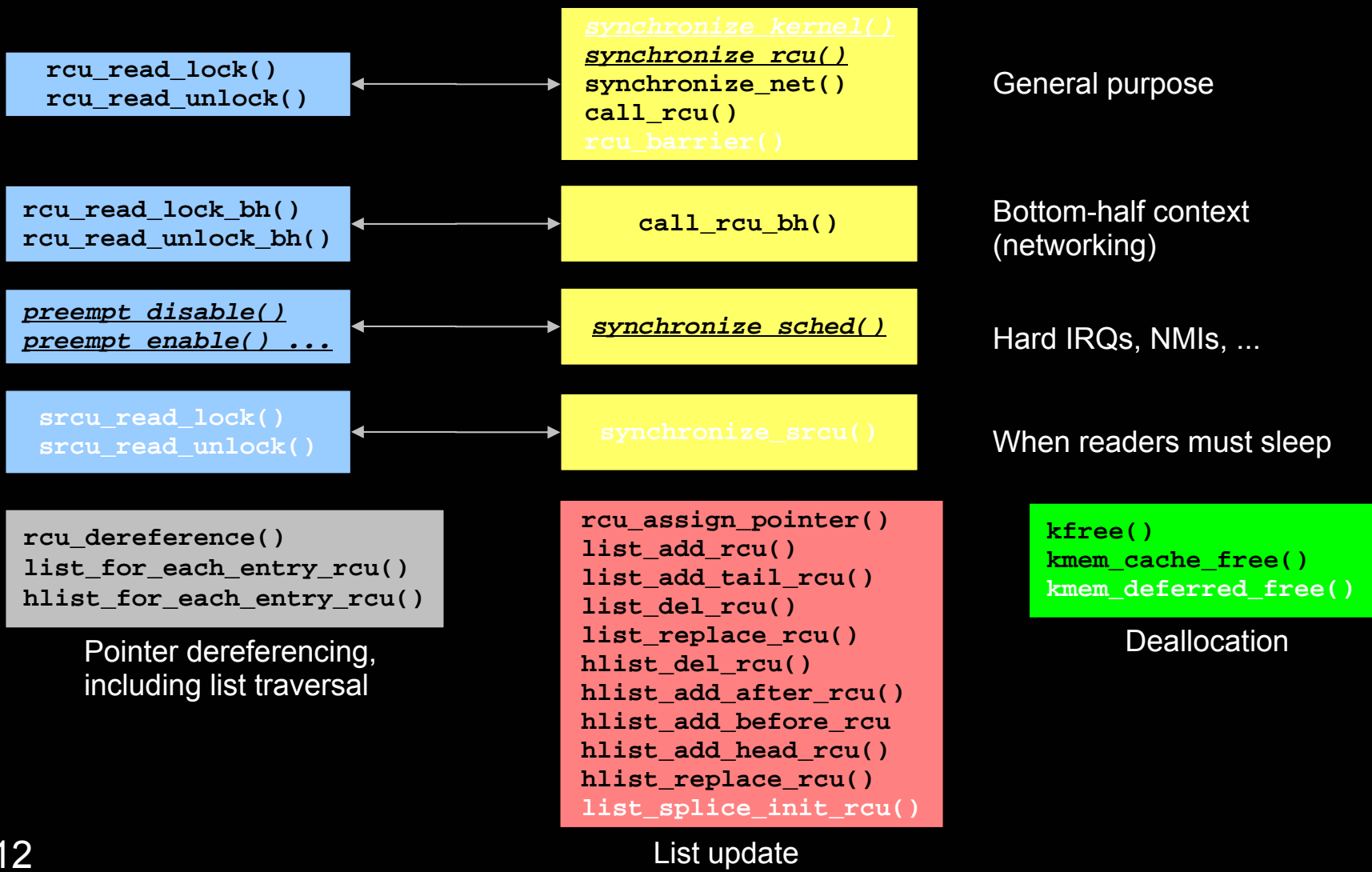
# Linux RCU函数的最终名称和实现



## Linux 内核实时操作系统（二）

- **People use RCU for its side effects**
  - For example, waiting for interrupts and NMIs handlers to complete
- **This makes it hard to implement an aggressive realtime implementation of RCU**
  - So we create an alternative API specifically for waiting for interrupt handlers and NMIs
  - See <http://lwn.net/Articles/134484/> for details

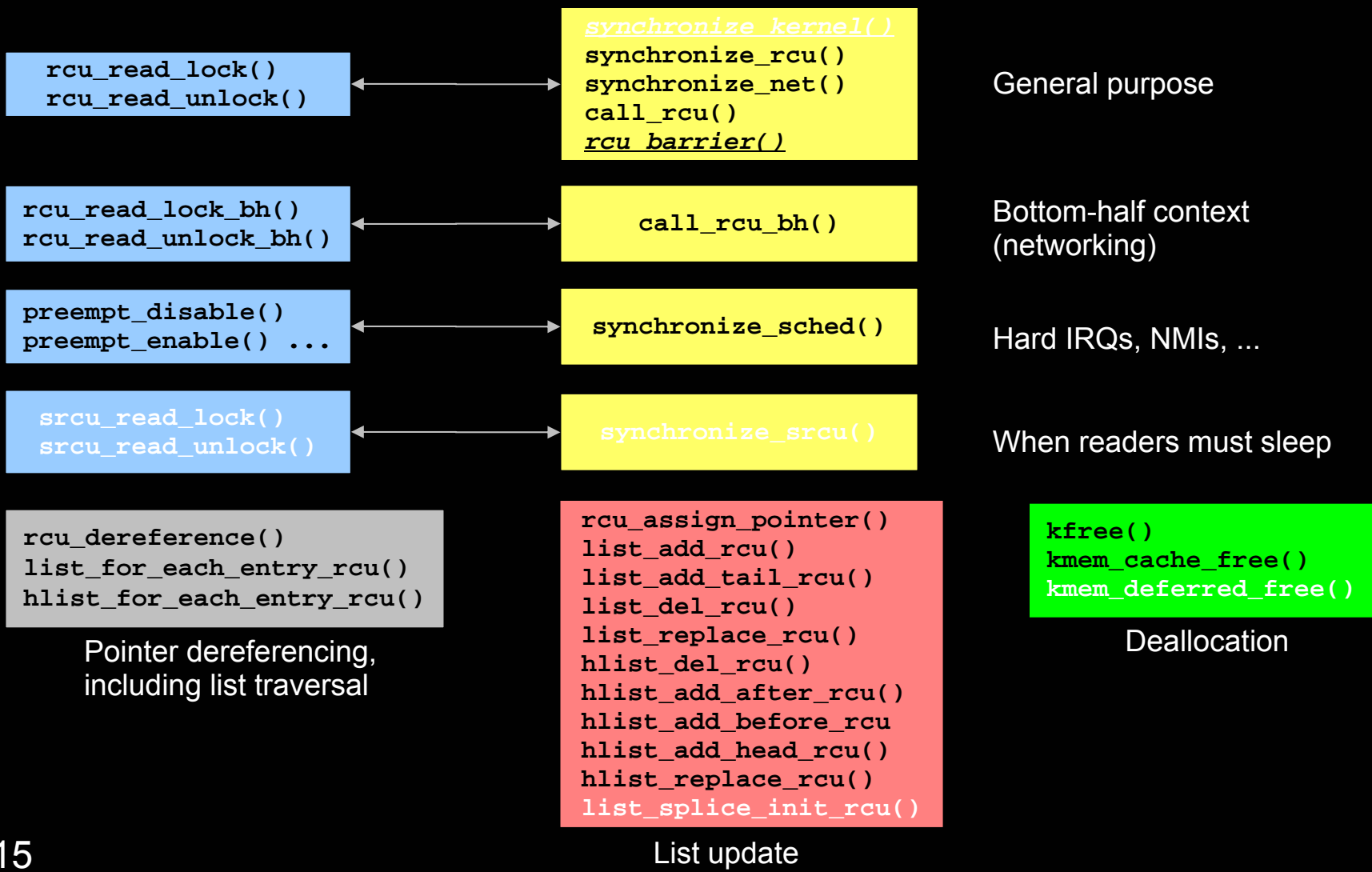
# Linux 内核实时操作系统 (二)



## 不可载入模块中Linux 内核对 RCU的使用

- **A given module's RCU callbacks can execute after a module is unloaded**
  - So that the affected callbacks cannot find their object code
  - See <http://lwn.net/Articles/217484/> for details
- **Added Dipankar Sarma's rcu\_barrier() primitive to allow a module to wait for all of its RCU callbacks to complete**

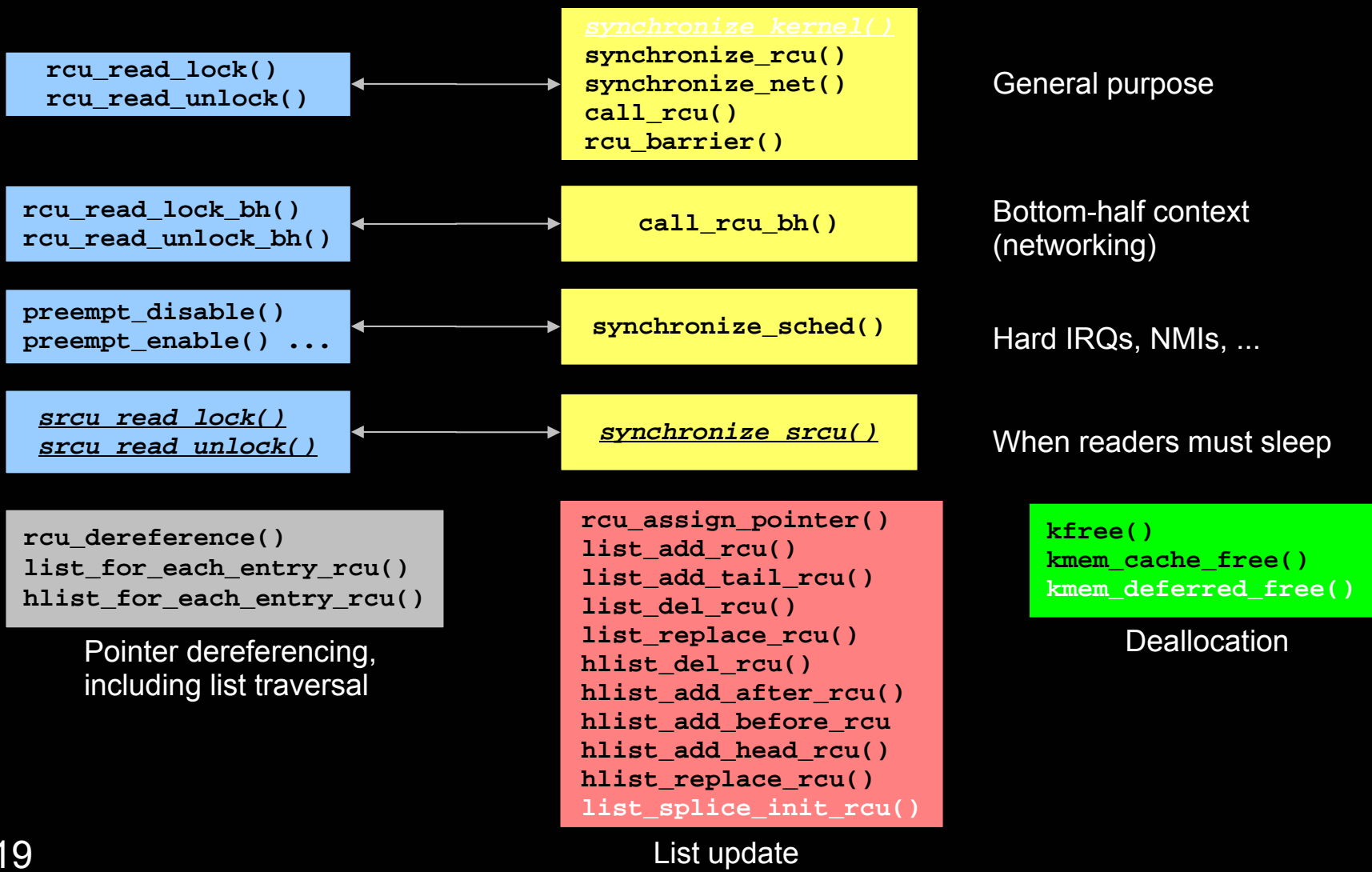
# 不可载入模块中Linux 内核对 RCU的使用



## Linux 内核中RCU 读取器需要休眠

- **For more than a decade, “I need my RCU readers to be able to sleep” meant that the speaker didn't really understand RCU**
- **Until 2006, when I found someone who really did need RCU readers to sleep**
- **Hence SRCU...**
  - See <http://lwn.net/Articles/202847/> for more details

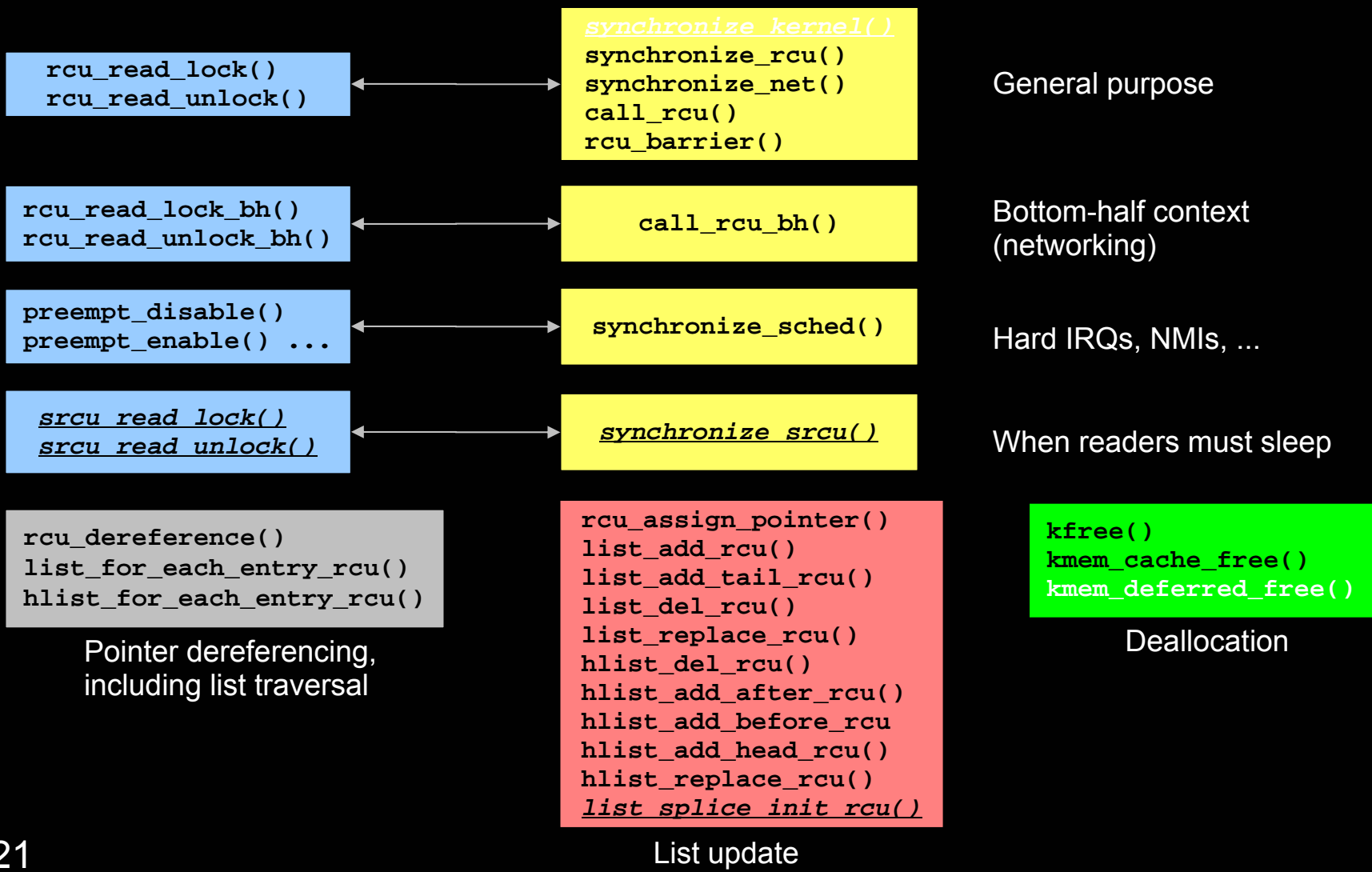
# Linux 内核中RCU 读取器需要休眠



## Linux 内核进行成熟的链表运算

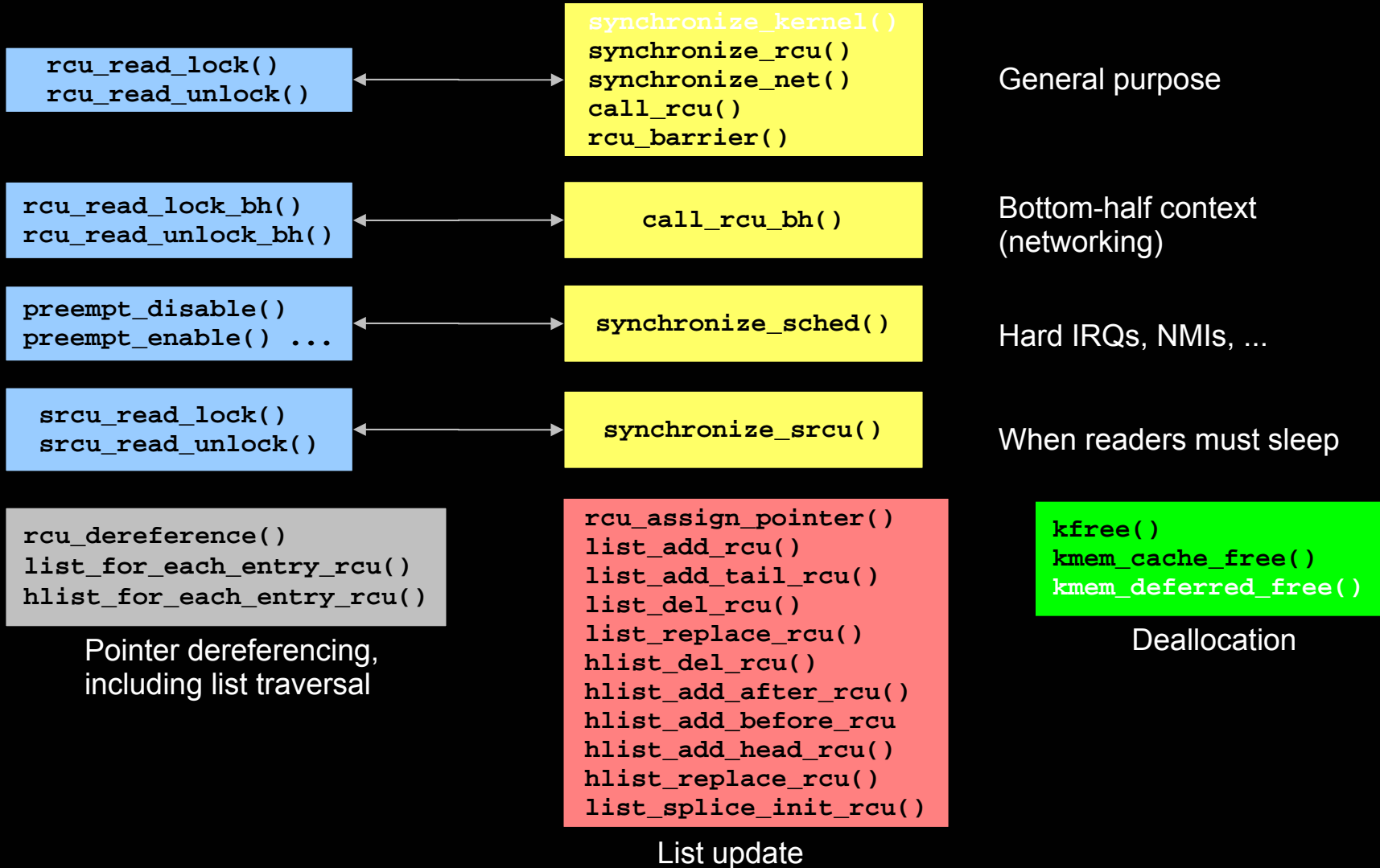
- **Reap an entire list while being traversed by RCU readers**
- **We were going to open-code it, but Christoph Hellwig made us create a primitive for this situation**
  - Corey Minyard does the heavy lifting

# Linux Kernel 进行成熟的链表运算



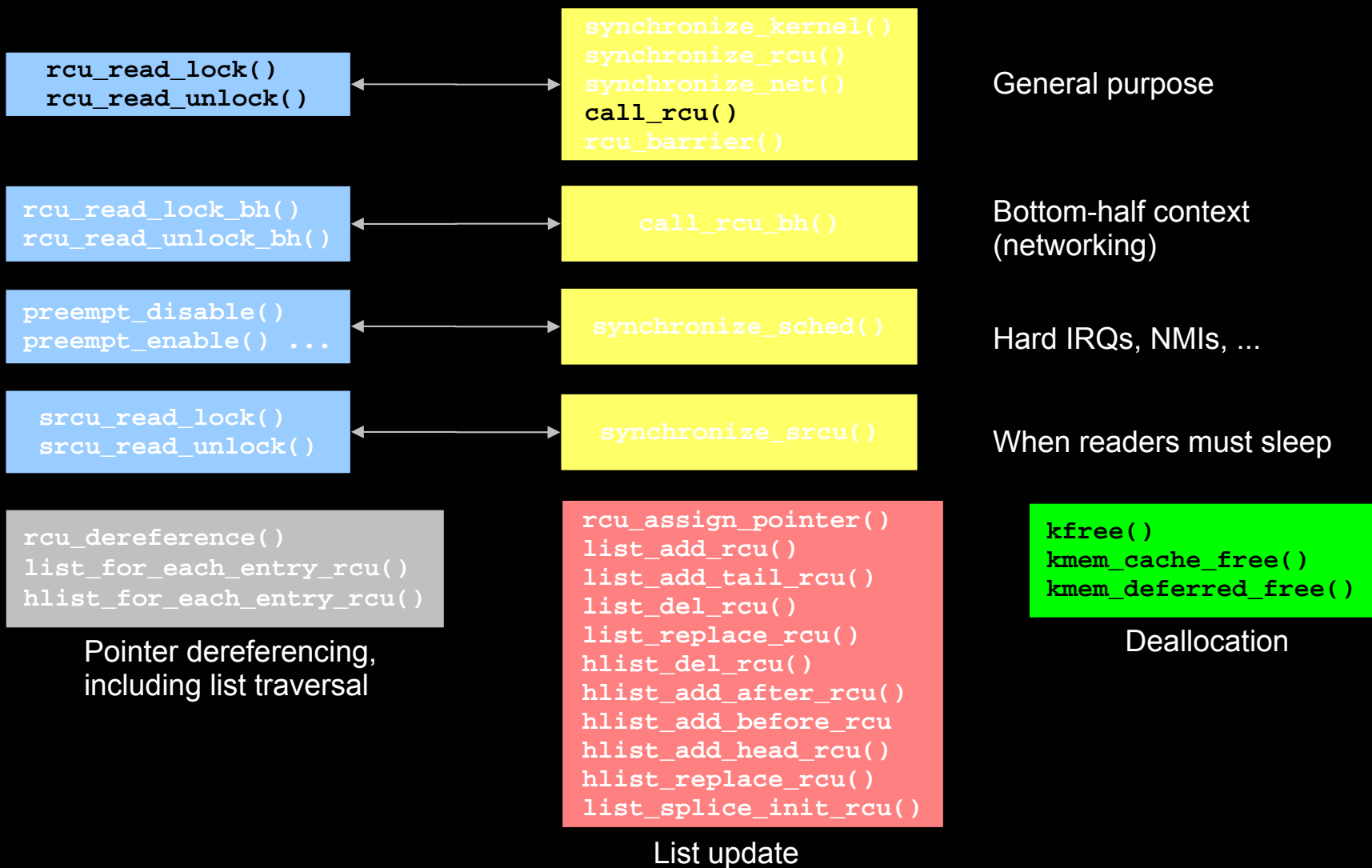
# RCU 的现状?

# Linux RCU API 2.6.24

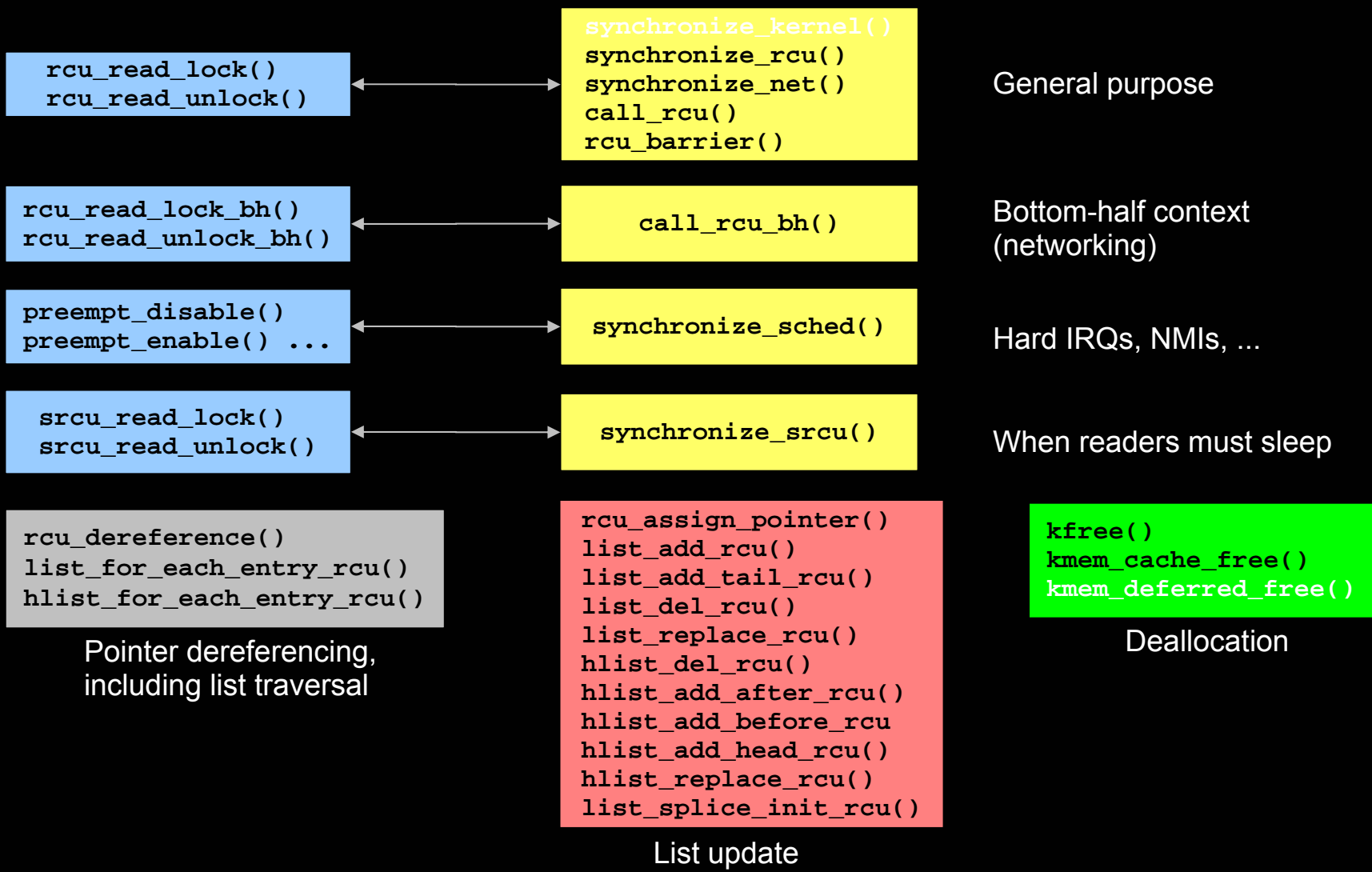


# 关于Linux 如何改变RCU的总结

# 这是你的技术



# 这是你的技术 : 6x API 增长



# 关于RCU 的经验总结

# 关于RCU 的经验总结

- **Linux** 可以运行以难以置信的种类的工作
  - 嵌入式、实时、桌面、网络、服务器和超级计算机 ...
- **Linux** 为重要网络基础结构提供强大支持
  - Linux 可被看作防火墙; 没被保护但胜过防火墙
- **Linux** 运行实时工作负荷
  - 实时的影响普遍深入
- **大量内核开发者情况 (数千)**
  - If 一人每年节省1% 时间:
    - ▶ Linux: ~10,000 开发者~每年节省 100 人-年
      - 不到四天便回收成本
      - 即使只有 500 全职开发者, 意味着十周即可实现盈利
    - ▶ 所有者: ~40 开发者 ~每年节省0.4 人-年
      - 两年多可以收回投资成本
- **在受保护环境开发出来的技术需要更多的修改!!!**
  - 为 Linux 引入技术是一项有益的学问

# 结论

为 Linux 贡献技术及其有益

*但不轻松!!!*

## 法律声明

- 本文表达了作者观点但并不代表**IBM**观点.
- **IBM, IBM (标识), e-business (标识), pSeries, e (标识) server, xSeries** 等是国际商业机器公司在全球的注册商标.
- **Linux** 是**Linus Torvalds**的注册商标.
- 其他公司、产品和服务的名称可能是其他公司的注册商标.

# 问题?