IT架构必知的定律

Huayulei_2003@hotmail.com
2019/01/15

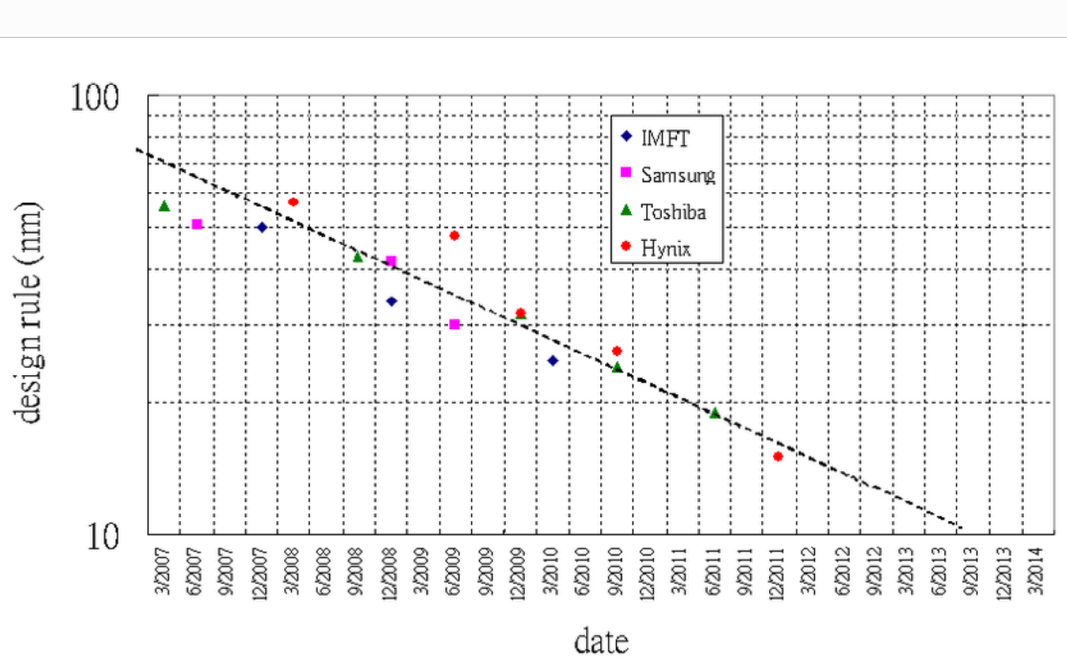
目录

摩尔定律 (Moore‘s Law)

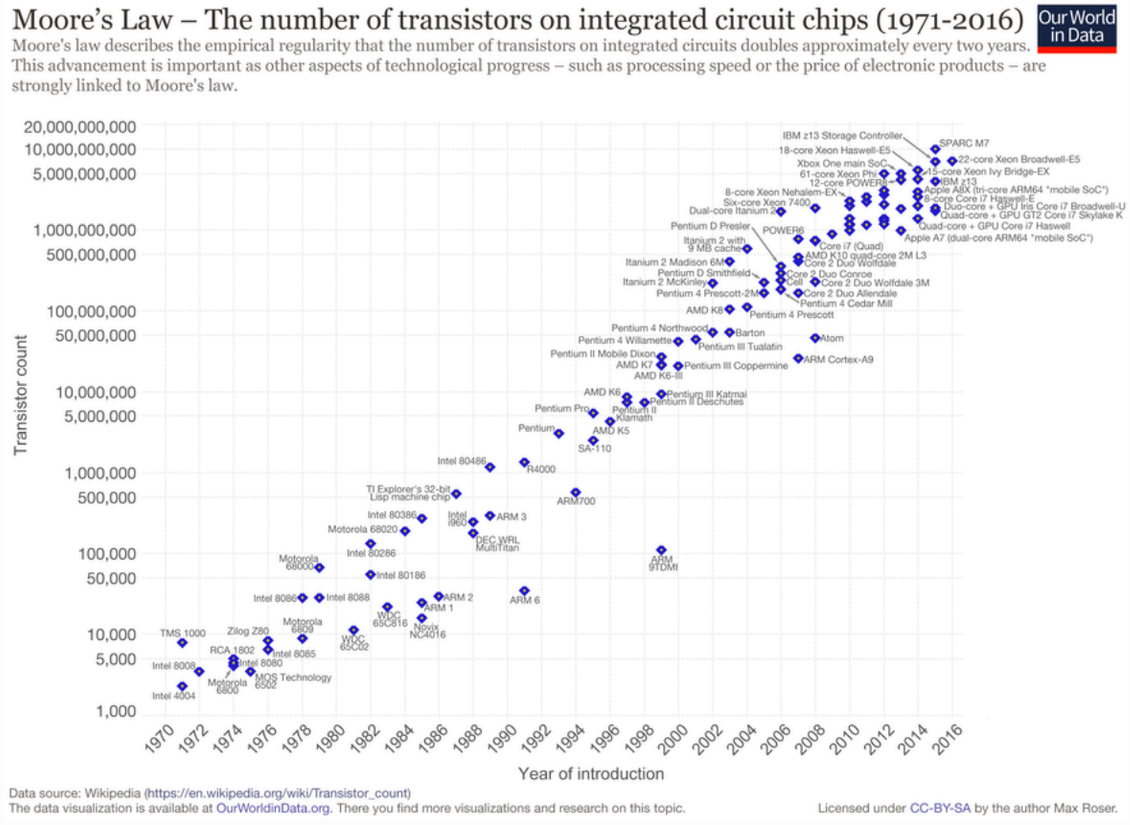
内容描述：

Moore's law is the observation that the number of transistors in a dense integrated circuit doubles about every two years.

“集成电路的晶体管密度大约每两年翻一番”



The trend of scaling for NAND flash memory allows doubling of components manufactured in the same wafer area in less than 18 months.



摩尔定律补充

摩尔第二定律：

Rock's law or Moore's second law, named for Arthur Rock or Gordon Moore, says that the cost of a semiconductor chip fabrication plantdoubles every four years.

“半导体芯片制造工厂的成本每四年翻一番。”

反摩尔定律:

谷歌前CEO埃里克·施密特提出的，“如果你反过来看摩尔定律，一个IT公司如果今天和十八个月前卖掉同样多的、同样的产品，它的营业额就要下降一半”

安迪-比尔定律:

安迪-英特尔公司CEO安迪·格鲁夫（Andy Grove）；比尔-微软公司创始人比尔·盖茨。 “Andy gives, Bill takes away.”意思是硬件提高的性能，很快被软件消耗掉了。

安迪-比尔定理把原本属于耐用消费品的电脑、手机等商品变成了消耗性商品，刺激着整个 IT 领域的发展。

康威定律 (Conway's Law)

内容描述：
organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations.

缘由：

第一定律：
Communication dictates design 组织沟通方式会通过系统设计表达出来

第二定律：
There is never enough time to do something right, but there is always enough time to do it over.
时间再多一件事情也不可能做的完美，但总有时间做完一件事情

第三定律：
There is a homomorphism from the linear graph of a system to the linear graph of its design organization
线型系统和线型组织架构间有潜在的异质同态特性

第四定律：
The structures of large systems tend to disintegrate during development, qualitatively more so than with small systems
大的系统组织总是比小系统更倾向于分解

康威定律被视为微服务架构的理论基础

墨菲定律 (Murphy's Law)

内容描述：

Murphy's law is an adage or epigram that is typically stated as: "Anything that can go wrong will go wrong".

墨菲定律是一句格言或警句，通常被表述为：“**任何可能出错的事情都会出错**”。

历史起源：

墨菲是美国爱德华兹空军基地的上尉工程师。1949年，他和他的上司斯塔普少校，在一次火箭减速超重试验中，因仪器失灵发生了事故。

墨菲发现，测量仪表被一个技术人员装反了。由此，他得出的教训是：**如果做某项工作有多种方法，而其中有一种方法将导致事故，那么一定有人会按这种方法去做。**

墨菲定律的适用范围非常广泛，它揭示了一种独特的社会及自然现象。

它的极端表述是：

如果坏事有可能发生，不管这种可能性有多小，它总会发生，并造成最大可能的破坏。

布鲁克斯法则(Brook's Law)

内容描述：

在《人月神话》中，布鲁克斯博士提出了布鲁克斯法则：向进度落后的项目中增加人手，只会使进度更加落后。

缘由：

在软件项目进度滞后的情况下，增加人手意味着可能要分配额外的资源来对新加入的人员进行培新和指导，也意味着人与人之间的交流变得更为复杂，在有N个人的项目中，两两之间的交流路径有（N^2-N）/2条。在N个人的项目中增加一个人，其交流路径条数便增加N条。

特殊情况

考虑以下几种特殊情况：

1. 工作比较简单，无需进行培新，或者是培训的时间可以忽略不计
2. 新加入的员工本身就是熟练工
3. 交流不频繁
4. 相对于增加人手所带来的培训、指导、交流的代价，其仍然提高了开发进度

霍夫斯塔特定律 (Hofstadter's Law)

内容描述：

It always takes longer than you expect, even when you take into account Hofstadter's Law.

“即使你考虑到霍夫斯塔特定律，它所花费的时间也总是比你预期的要长。”

缘由：

霍夫斯塔特在讨论国际象棋计算机时引入了这一定律。尽管在递归分析的深度上超过了人类，但在当时，国际象棋计算机一直被顶级人类棋手打败。霍夫斯塔特写道**“在计算机国际象棋的早期，人们常常估计要过十年计算机(或程序)才能成为世界冠军。但十年过去了，计算机成为世界冠军的日子似乎离我们还有十多年……”**

特殊情况

1. 值得注意的是，1997年深蓝击败加里•卡斯帕罗夫的那一天确实到来了。这表明，当任务重复时，加速回报定律可能会生效，从而抵消霍夫斯塔德定律，在某些情况下甚至推翻霍夫斯塔德定律。(推论)
2. AlphaGo击败人类职业围棋选手、战胜围棋世界冠军。 2016年3月，阿尔法围棋与围棋世界冠军、职业九段棋手李世石进行围棋人机大战，以4比1的总比分获胜； 2017年5月，在中国乌镇围棋峰会上，它与排名世界第一的世界围棋冠军柯洁对战，以3比0的总比分获胜。

伯斯塔尔定律(Postel's Law)

内容描述：

Be conservative in what you do, be liberal in what you accept from others (often reworded as "Be conservative in what you send, be liberal in what you accept").

在你所做的事情上要保守，在你从别人那里接受的东西上要自由(经常被改写成“在你发送的东西上要保守，在你接受的东西上要自由”)。

在计算中，鲁棒性原则是软件的设计准则。

缘由：

该原则也被称为Postel定律，以Jon Postel的名字命名，他在TCP的早期规范中编写了该定律

帕累托法则 (Pareto Principle)

内容描述：

帕累托法则(也被称为80/20法则，重要的少数法则，或因子稀疏原则)指出，对于许多事件，大约80%的影响来自20%的原因。

意义：

在计算机科学中，帕累托原理可以应用于优化工作。例如，微软指出，通过修复报告最多的前20%的bug，可以消除给定系统中80%的相关错误和崩溃。[15] Lowell Arthur表示：“20%的代码有80%的错误。找到他们，修理他们!此外，研究人员还发现，一般来说，某个软件的80%可以在总分配时间的20%内完成。相反，最难的20%的代码占用了80%的时间。这个因素通常是COCOMO估计软件编码的一部分。

90-90法则 (Ninety-ninety rule)

内容描述：

The first 90 percent of the code accounts for the first 90 percent of the development time. The remaining 10 percent of the code accounts for the other 90 percent of the development time. — Tom Cargill, Bell Labs

“前90%的代码占用了前90%的开发时间。剩下的10%的代码占用了另外90%的开发时间”

意义：

90%+90%合起来是180%，表达的是软件开发项目显著地超出了它们的时间表。它既表达了对编程项目中容易部分和困难部分的粗略时间分配，也表达了许多项目延迟的原因，即没有预见到困难部分。换句话说，完成一个项目所花费的时间和代码都比预期的要多。

林纳斯定律 (Linus's Law)

内容描述：

The law states that "given enough eyeballs, all bugs are shallow"; or more formally: "Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone."

“足够多的眼睛，就可让所有问题浮现”或者“只要有足够的测试人员及共同开发者，所有问题都会在很短时间内发现，而且能够很容易被解决”

意义：

被公开审查、测试的代码越多，各种形式的错误就能更快地被发现。

1. 深入讨论对比大教堂和集市开源软件开发模型
2. 对代码review和代码测试方法有一定的指导意义

总结

1、在系统设计时，应该多思考“墨菲定律”

2、在系统划分时，也要思考“康威定律”

3、在工期估算时考虑“霍夫斯塔特定律”

4、在功能设计时考虑“伯斯塔尔定律”

5、在变动项目安排时，考虑一下“布鲁克法则”

6、在分析问题时考虑“帕累托法则”，又称80/20法则

7、在控制代码质量时，多考虑“林纳斯定律”

8、在系统优化时，多考虑“克努特优化原则 (Knuth's optimization principle)”

Q&A

Thank you very much !