故障预测技术研究综述

● 严 然 孟 由 钱德沛 栾钟治
北京航空航天大学计算机学院 北京 100191

摘要：

随着计算机系统体系构造的日趋复杂、计算规模的不断扩大，系统可靠性受到了严重影响。主动容错技术是提高系统可靠性的一个有效手段。故障预测作为主动容错技术的一个重要方面，研究如何通过监控系统当前的状态并判断系统是否即将发生失效进而提高系统的可靠性。本文首先给出故障预测的相关定义以及评估预测方法性能的标准；其次，依据分析对象的差异对故障预测技术提出一种分类方法，在分类的基础上，介绍了各种典型的故障预测方法并进行了对比；最后对故障预测的相关技术进行了分析对比总结，并讨论了未来的研究趋势。

关键词：故障预测，系统可靠性，预测评估标准，容错，监控

引言

由于计算机系统日渐庞大，构造日趋复杂，加之组件的动态增减，运行环境变更，系统在线升级和在线修复工作频繁，以及网络环境自身引入的复杂性都导致系统可靠性的下降。今天在大规模并行系统中，故障已经成为一种常态，很多大规模并行系统平均每2天就要经历一次故障[1][2]。依照这种故障频率，未来的更大规模的E级(Exabyte Flops)众核系统的平均故障时间（Mean Time To Failure，简称MTTF）甚至可能缩短到几小时。

另外，随着用户对在线应用依赖程度不断增加，支撑相应服务（例如：云计算服务）的平台规模日益增大。Google等公司构建包含几万甚至几十万个节点的基础设施来支撑云计算平台的日常运营，这类基础设施容易受到软硬件故障的影响，使得节点不可用现象频繁出现。相应地，其基础设施及服务的可用性也受到了极大的关注。系统规模不断增大导致了MTTF不断缩短，例如：IBM的ASCI White，在不断提高可靠性的基础上，平均无故障时间（Mean Time Between Failure，简称MTBF）时间仅为40小时左右[3][4]；根据预测，处理器数目超过100,000的IBM Blue Gene-L的MTTF可能只有几十分钟[5]；Google分析了其部署在不同地域的几十个包含1000到8000不等节点的站点一年内运行数据指出，每年的结点故障率为2-3%，也就是说每36小时就会有一个结点发生故障[4]。运行在平台之上的应用由于软件错误等原因也会影响系统可用性，例如：MapReduce发现，数据错误、程序BUG、磁盘I/O故障等原因会导致计算出错或长时间不返回结果[7]。另外，平台日常管理、维护及升级有时候也会导致系统不可用现象出现。因此，从整体角度讲，大规模基础设施比单个节点更容易受到故障的影响，在这种场景下，故障行为已经逐渐成为一种常态。因此，大规模并行环境下系统的可靠性保障变得至关重要。

用于保障系统可靠性的研究已经有很多，传统的可靠性理论以及众多通用的可靠性方法往往根据普遍经验得到通用模型，按照其针对故障处理的方式不同大体可分为故障检测，故障预测，以及故障恢复策略。故障检测用于在程序运行过程中查询系统是否产生了故障，并及时通告结果以采取相应措施，从而减少故障带来的损失。故障恢复是在故障已经发生并且给系统带来不可避免的影响时采取的应对措施，其目的是将程序恢复到正确的运行轨迹当中。而故障预测则是基于对系统历史状态以及当前行为的分析，判断系统是否即将产生故障，协助系统规避故障或者尽早采取故障恢复措施。

较之其他两种保障系统可靠性的方法，故障预测可以在故障出现之前，通过分析系统的状态得到故障可能出现的概率以及可能出现的故障种类。一方面，用户通过故障出现的概率以及出现的位置，采取相应的手段避免这些有可能出现的故障；另一

方面，当故障出现概率已经很大甚至不可避免的时候，也可以更早地主动采取相应的恢复措施，而非等到故障完全出现再进行恢复，从而能够在提高系统可靠性的同时最大限度地减少系统因故障处理而带来的资源损耗。可见，故障预测技术是一种有效的主动故障处理技术。

针对故障预测，研究者已经从不同的角度、结合不同的手段，开展了广泛的研究工作。本文对面向计算机系统的故障预测技术进行综述，并指出未来研究的方向。本文第1节首先给出故障、差错和失效等相关概念的定义，进而对故障预测的含义进行详细阐述。第2节针对故障预测的准确性以及正确性给出度量标准。第3节提出面向计算机系统的故障预测技术的分类框架。第4节结合分类框架，详细分析了目前常见的故障预测技术，对在第5节进行了总结比较。第6节对未来的研究工作进行展望。第7节总结全文。

1. 相关定义

为了使本文中所涉及的故障预测的概念被更加清晰地地表述，我们在这一节里首先对其涉及到的一系列术语给出定义。

1.1 影响系统可靠性的几个因素

一直以来，从事此领域研究的工作者都在试图就系统可靠性受到影响的程度划分出状态，例如：Melliar-Smith[8]、Avizienis[9]、Laprie[10]以及Siewiorek [11]都在自己的著作中对此问题做出论述。但是目前得到广泛认可的是Avizienis等随后在[12]提出的观点。他认为影响系统稳定性的原因通常可以按照发生的先后顺序可划分为故障（fault）、差错(error)和失效(failure)。

故障通常是指硬件在物理上的不完整或缺陷以及软件组件在编写时逻辑漏洞等。它是产生系统不稳定因素最根本的原因。多数情况下系统故障不会被触发，但是它一旦被触发便会引起差错的产生。故障一般又分为瞬时故障、间歇性故障和永久性故障[11]。

差错指系统运算结果不正确或不精确导致统状态偏离的正确的状态。因此可以说差错的发生造成系统的局部失效，但这种状态尚未被用户感知。

失效发生在系统级，指系统在一个特定时间段内提供的服务与其预期不相符。这里的关键点在于系统的错误行为是可以被用户或者其他设备所感知，也就是说系统内部尚未造成错误输出的差错便不能称之为失效。

在以上观点的基础上，F.Salfner等在[13]中对上述划分过程的进行了补充和扩展。首先他们将系统差错分为“未被检测到的差错”以及“尚未被检测到的差错”。同时，他们认为虽然差错状态是无法被用户所感知，但是它必然会对系统产生一些边界效应。例如，内存泄漏往往不会立刻导致系统失效，系统首先会因为Swap Memory不足变得运行缓慢，消耗内存量持续增加，这些状态即为边界状态。他们将这里的边界效应称为“差错症状”。图1[13]给出了影响系统可靠性的几类因素以及其演变过程。

1.2 故障预测

预测技术被应用到众多行业，包括航空，气象，政治，股票分析师以及IT行业等，针对系统故障的预测最早可以追溯到30年之前[22]。故障预测可以在故障发生之前就对故障做出反应，作为故障检测一种补充手段，故障预测直到最近10年左右才逐渐被学术界以及工业界重视。许多预测相关领域的研究被广泛的开展，包括自动计算(Automatic Computing)[23]，可信计算[24]，可恢复计算[25](Recovery-oriented Computing)等，这些领域关注的问题相对类似：如何让系统能够尽可能积极的的处理故障，即更早的对故障进行处理。其研究范围广泛，涉及计算模型，分类方法，安全策略以及机器学习等[26]。

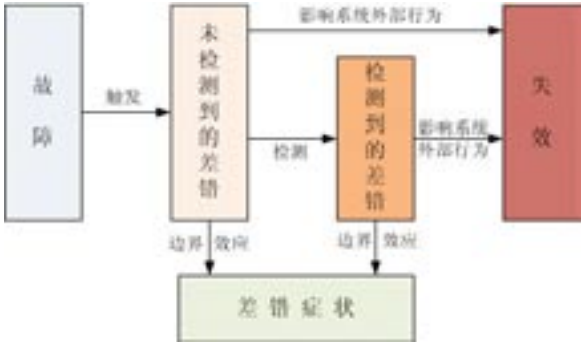

图1 影响系统可靠性的因素以及其演变过程

故障预测作为一种主动容错技术是一种使用抽象的设施或组件，通过分析系统当前的状态进而判断系统是否即将发生失效。它可以避免故障以及故障修复带来的开销，能够有效抵御故障传播带来的影响，具有容错开销小，处理过程简单，对用户透明等显著优势。

故障预测实现的基本思路如图2所示，横坐标轴表示系统时间，其正方向是时间流逝的方向[13]。当前系统处在时间t，故障预测基于系统当前状态以及历史状态（图中长度为Δt_d的时间范围），分析并预测在Δt_l时间之后系统可能出现的故障。同时，由于预测结果具有时间维度的要求，预测方法需要给出

所预测的故障发生的时间范围，在图中记为Δt_p。此外，时间段Δt_w表示系统针对一次故障处理需提前准备的时间，例如提前进行检查点备份等，因此故障预测的时间Δt_l必须大于Δt_w，系统才能有足够时间准备应对一次故障。

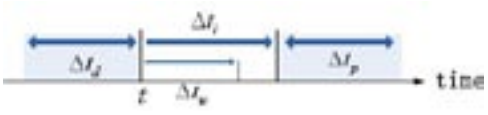

图2 故障预测中的几个基本约束时间

2. 故障预测方法的评判标准

为了比较各故障预测方法的性能优劣，采取一套科学合理的评判标准是十分必要的。在这一节里我们主要介绍一种适合故障预测方法的评判标准。

在故障预测系统中，衡量故障预测方法性能的指标主要包括预测的正确性以及预测的准确性两个方面。预测的正确性是指故障预测算法要尽可能少的产生虚假报警。预测的准确性则是指故障预测算法覆盖尽可能多的系统故障。预测的正确性与准确性两个标准的关系可直接通过图2中的故障发生时间Δt_l以及故障发生窗口Δt_p进行反映。由于Δt_l用于表示提前预测到故障发生的时间，因此该值越大表示的准确性越高；故障发生窗口越小，则预测的故障发生时间范围越小，因此预测越准准，相反当该预测时间段$\Delta t_p \to \infty$时，只要出现故障，该预测一定为正确，但是该预测也因此变得毫无意义。理想的故障预测算法是做到预测结果与真实的系统故障一一对应。

关于预测方法的性能表述，研究者已经就此提出了一些方法。例如：Atlman就在[14]中基于kappa统计量的方法对预测性能进行表达。但是，这种方法很少被该领域的研究人员所使用。目前大部分对预测方法的评估工作都是建立在如下设定的基础上。

假设在实际使用故障预测模型的过程中，会产生四组正确性相关的数据。如果预测算法判定在某时间段即将出现一个故障，则可表述为一个故障事实(positive)，在该时间段内，可能真的发生了故障(TP : true positive)，也可能没有发生故障(FP : false positive)，即误报。同时，在预测模型断定不会发生故障(negative)的时间段，可能真的没有故障发生(TN: true negative)，也可能发生了故障(FN : false negative)，即漏报，四种预测结果如表1所示[15]。

表1 预测结果列连表

	True failure	True nofailure	Sum
Prediction:failure (failure warning)	True positive(TP) (correct warning)	False positive(FP) (false warning)	Positives (POS)
Prediction:no failure (no failure warning)	False negative(FN) (missing warning)	Ture negative(TN) (correctly no warning)	Negatives (NEG)
Sum	Failures(F)	Nonfailures(NF)	Total(N)

基于上述对预测结果的设定，Weiss于[15]中采用Precision/Recall模型对预测的结果进行评估。在Precision/Recall评估模型中，Precision用于表示某模型预测的结果中(包括预测出现故障以及预测不出现故障两种)，正确的预测结果所占的比例，它代表了预测的正确性，即：

$$Precision=\frac{TP}{TP+FP} \quad (1)$$

而Recall则表示模型预测到的真实发生的故障占该环境中发生的所有发生的故障的百分比，它代表了预测的准确性，即：

$$Recall=\frac{TP}{TP+FN} \quad (2)$$

例如，通过评测某个预测算法的Precision达到0.78，即故障预报的正确率为78%，误报率为22%；同时它的Recall达到0.91，则表示在系统运行过程中有91%的故障被预测出来，有9%的故障被漏报。

通过Precision/Recall模型的定义我们可以看到，如果要提高算法的正确性（Precision），需要降低误报（FP）事件的发生。但是，如此以来则有可能造成算法对故障的预测覆盖率下降，即导致准确性（Recall）的下降。为了表现出预测算法在正确性和准确性方面综合的性能，Van Rijsbergen在[16]中提出了调和平均数F-measure。在Precision和Recall权重一致的情况下，F-measure定义如下：

$$F\text{-}measure=\frac{2\cdot precision\cdot recall}{precision+recall}\in[0,1] \quad (3)$$

对于正确性和准确性同时具有很好性能的预测算法，F-measure的值也会很高；反之，随着Precision或Recall的值趋于0，F-measure的值也会趋向于0。

此外还有相关研究使用精确性(Accuracy)，其表示预测模型正确判断的事件占总共事件的百分比，其中，事件包括“可能发生故障”以及“不会发生

故障”两种，其表述为：

$$Accuracy=\frac{TP+TN}{TP+TN+FP+FN}=\frac{TP+TN}{N} \quad (4)$$

由于在实际的系统运行过程中“系统失效”事件的发生具有不确定性，而且相对于“系统正常运行”事件来讲，“系统失效”应当是发生概率较低的事件。所以在利用上述指标对某个预测算法进行性能评估的时候，我们需要一个参考数据集来明确“系统失效”事件的发生以及其发生的时间。这里的参考数据集类似于机器学习中的：Labeled Data Set[17]。同时，这个数据集应当足够的大，以便对TP、FP、FN以及TN等指标事件进行充分的统计。

在实际评估过程中，参考数据集可以分为训练数据集和测试数据集两个部分。预测算法可以利用训练数据集进行自学习进而优化算法参数。测试数据集则是被用来对预测算法进行评估。同时为了评估结果的更准确，在对TP、FP、FN以及TN等指标事件判定的过程中，对预测时间段Δt_p的设置应当根据系统故障管理的整体要求进行设定。

通过上述测量手段得到评估结果后，由于参考数据集的有限性以及指标事件统计过程中的误差，我们还应针对评估结果给出其相应的置信区间。这里对置信区间的确定涉及到交叉检验、Bootstrap、Jackknife等统计学方法，Salfner在[18]中对这些方法有详细的介绍。

3. 故障预测技术分类框架

目前故障预测技术依据分类的方法很多，可以有不同的分类方式。在预测过程中，没有对实际情况以及具体环境的特性进行分析就盲目选择预测方法，会严重影响预测模型的准确性，导致故障预测准确性低、覆盖面不高。这一节中我们参照1.1节中对影响系统可靠性因素的定义，根据预测方法分析对象的不同大体可将故障预测技术划分为基于故障历史的故障预测技术、基于状态监控的故障预测技术、基于事件驱动的故障预测技术以及基于校验的故障预测技术四类。图3所示为本文提出的分类框架。

（1）基于故障历史的故障预测技术主要分析的对象是引发“系统失效”的系统历史数据。这类技术的基本思想是通过对引发系统失效的历史数据的分析总结出历史故障与即将系统发生的故障之间的规律进而进行预测判断。根据实现技术的差异，又可分为基于故障历史概率分布的故障预测和基于故障重现假设的故障预测。此类预测方法可以对系统即将发生故障的时间及类型进行预测。

图3 故障预测技术的分类

（2）基于状态监控的故障预测技术主要监控的对象是由“系统差错”的边界效应所带来的“差错症状”。正如1.1节中提到的诸如内存泄露一类的系统差错在引发系统失效前是不会被检测到的，但是我们可以通过对系统的内存使用、CPU负载、磁盘读写、网络I/O以及系统函数调用等可能受到系统差错影响的参数进行异常检测来确定系统是否处于差错状态，进而对即将发生的系统失效进行预测。根据检测技术的不同，相应的又可分为基于函数模拟的故障预测、基于分类的故障预测以及基于系统模型的故障预测三种方式。这类预测方法通常被用来解决non-fail-stop[18]失效的问题。

（3）基于事件驱动的故障预测技术主要分析的对象是系统针对“检测到的差错”反馈的事件数据。系统一旦检测到差错事件的发生，它会以日志或者消息的方式通知用户。因此，通过对此类事件数据的学习和分析，可以对即将发生的系统故障进行预测。此类故障预测方法一般都涉及到模式识别技术。根据识别策略的不同，此类方法可以进一步分为基于规则的故障预测以及基于事件组合序列的故障预测。

（4）基于校验的故障预测技术主要针对系统中“未检测到差错”这类情况所提出的。为了尽可能早地对系统故障进行预测，我们可以通过对系统相关数据的一致性检验，来达到检测差错进而预测故障的目的。这类技术在对运行时的系统故障预测中使用的比较少，所以在本文中没有对它做进一步分类。

4. 故障预测技术

4.1 基于故障历史的故障预测方法

基于故障历史的故障预测方法全部基于如下假设：系统已经被正常配置，程序也是可以正确执行的(如果不能正确执行则为错误，而非故障)，因此系统出现故障往往是因为存在外界扰动，此类扰动必然使系统存在时间以及空间的周期性变化。导致故障的发生在时间以及类型上必然存在某种周期性，该方法则利用该特性对即将发生的故障进行预测。

4.1.1 基于故障历史概率分布的故障预测

基于故障历史概率分布的故障预测模型试图通过对历史故障时间间隔进行分析得到即将发生的故障在时间上的概率分布。通常此类方法会在系统非运行时利用故障历史信息对未来故障的发生进行可靠性预测，并将分析结果运用于系统运行时的故障预测中。

基于以上思想，Aitchison等在[19]中提出了利用贝叶斯网络的预测方法。Csenki等[20]在其基础上进行了改进，提出了一种非运行时的、基于贝叶斯网络的故障预测方法。他们通过分析故障间隔时间（time-between-failures，TBF）的概率分布，对系统进行故障预测。此方法被用于对Jelinski-Moranda可信性模型[54]的性能改进，大大提高了该模型的预测准确度。

一些研究人员则发现，当系统处于故障过程时是不稳定的，因此系统TBF的概率分布也是不断变化。在系统实际运行中，系统的配置改动、升级、bug修复甚至是使用模式的变更等都有可能对故障过程产生影响。因此，他们认为以TBF稳定分布作为假设前提的技术都是不充分的。Pfefferman和Cernuschi-Frias[21]基于上述观点，将故障产生的模式假设是伯努利模型，既第k类故障在时间n发生的概率为$p_{k(}n)$。这样，求第k类故障的TBF分布的问题就转化为求“系统第n次输出结果为第k类故障”这一事件的发生概率，具体概率分布模型如下：

$$Pr(TBF_k(n)=m \mid \text{failure of type } k \text{ at } n) = p_k(n)(1-p_k(n))^{m-1} \quad (5)$$

基于此模型，他们在文中提出一种基于时间窗口的回归算法，可以对即将到来的指定类型故障的发生时间进行预测。

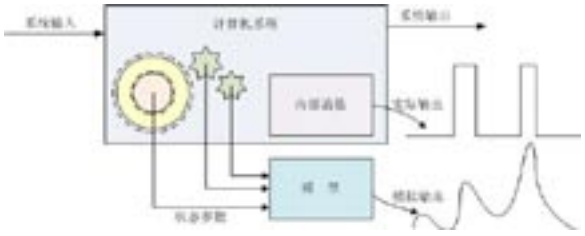

图4 基于函数模拟的故障预测模型

4.1.2 基于故障重现假设的故障预测

在许多共享资源（网络、存储等）系统中，故障的发生大多具有时间以及空间上的相邻特性。因此，此类预测方法多用于集群等资源共享型系统中。

Tang在[27]对DEC VAXcluster集群系统的故障数据进行分析，并给出故障模型。具体描述如下，假定集群拥有n个节点，E_i代表i个节点同时（时间粒度被选为1秒），例如，E_0代表系统目前没有节点处于故障状态。并基于此模型，进一步给出计算系统自状态i向状态j转变的故障状态转移概率p_{ij}，具体计算公式如下：

$$P_{ij} = \frac{\text{observed no. of transitions from } E_i \text{ to } E_j}{\text{observed no. of transitions out of } E_i} \quad (6)$$

Liang等人[28]基于以上的系统特性，针对IBM的BlueGene-L的日志分析了系统的可靠性、可用性以及服务性等特征。该工作亮点在于，他们在日志分析的过程中首先采用通过阈值的方式对日志数据采取基于时间以及空间的压缩，例如，某个节点附近的节点在某段时间窗口内发生的事件全部按照特定阈值压缩成单个事件，以便于以后的数据处理。基于上述基本假设，其预测方法相对直接：如果某种类型的故障已经发生了，下一个故障也可能是此种类型的故障或者相似的故障；同样，如果某个空间节点附近发生故障，同样下一个故障很可能依然发生在该节点附近。

Fu和Xu等人[29]在Liang的工作之上进一步阐述了基于时间以及空间的压缩方法，并就分布式环境下如何判定压缩后的时空数据是否为同一个故障数据等相关问题进行了改进。

4.2 基于状态监控的故障预测方法

此类方法目的在于通过监控系统状态的手段实现故障预测，如图1所示，系统从出现异常到失效是一系列事件联合作用的结果。其中，当系统出现异常后，会产生边界效应，该方法即通过探测边界效应来判定系统是否即将出现差错。当系统检测到边界状态，则说明系统即将演化出失效等问题，而后根据模型预测出故障产生的时间以及类型。此类方法的故障预测时间Δt_l较短，但故障发生时间窗口Δt_p也很短，因此准确性极高。

4.2.1 基于函数模拟的故障预测

此类方法将系统的状态参数作为输入，对系统内部的函数进行模拟。一般被用来模拟的目标函数对于故障预测者来说都是内部结构未知的。因此目标函数应当至少具备以下特点之一：

（1）函数发生故障的概率分布是可以获得的，这种情况下才能构建的模型并结合训练数据集对模型参数进行优化。操作时，用户可以将模拟的输出结果同实际的输出结果相比较进而判断系统是否即将发生故障，具体的如图4所示。

（2）函数所涉及的计算资源情况应当是被了解的，如“剩余内存”等相关系统变量等。此时模型可以对函数的资源使用情况进行推算。用户通过将推算结果与实际测量值进行比较来判断系统是否即将发生故障。

利用随机函数模型来进行故障预测，最早由Vaidyanathan等人[57]提出，该工作试图通过负载相关的输入数据（例如：系统函数调用的数量）获得某些相关的系统变量（交换内存以及实际空闲内存）。他们构建了一个半回馈的马尔科夫模型(semi-Markov reward model)以便于对负载相关数据进行估计。而后这些预测数据被用于分析系统资源何时将会耗尽，因为系统资源（如：空闲内存以及交换空间）的耗尽也意味着系统故障的产生。

Li等人[30]通过收集Apache Web服务器的各种性能数据，借助辅助手段(ARX)建立了可自回归的系统模型用于预测未来系统参数变化，而故障的预测则基于资源耗尽的时间。他们将自己的工作与Castelli等人[56]的工作进行比较并表明，在他们的数据测试集中，基于ARX的方式会比之前的方式效率更高。

神经网络技术作为机器学习的一种，目的也是通过对函数模型的启发式学习更好地完成系统功能模拟。Troudet等人[31]最早提出利用神经网络技术在机械故障预测方面进行研究。而后，Wong等人[32]较好地使用神经网络技术对阻抗无源原件电力系统进行模拟，并通过RLC-Π模型生成的数据集对预测模型进行训练。Neville则在[55]中详细介绍了神经网络技术在对大规模工程系统进行故障预测方面的相关问题。

Hoffmann等人在[33]中基于RBF(Radial Basis Function)模型扩展实现了UBF(Universal Basis Function)模型，该模型被广泛使用与网络与电信环境中的故障预测。而后，他们在进一步的工作[34]中，总结了前人通过模拟系统消耗资源预测故障的方法并在Apache Web服务器的故障预测中对其方法进行了实验验证。实验结果表明：在预测“剩余空闲内存”的过程中，UBF方法具有较高的准确性，而针对“Web服务器响应时间”的预测中，基于向量机的预测则具有更高的效率。

4.2.2 基于分类的故障预测

此方法类似于故障检测中的分类方法，该方法首先分析历史数据，根据数学分类方法将系统的状态分为有故障倾向的以及无故障倾向的两类，如图5所示。在系统运行过程中，该方法只需监控系统当前状态，若其状态被划分为无故障倾向分类，则判定短时间内不会出现故障；若当前状态被划分至故障倾向状态，则判定系统即将出现故障。但是由于该方法在确定分类的判定基线的时候是利用Labeled Data Set[17]作为训练数据集，所以其正确性对训练数据集的依赖较大。同时，由于该方法仅仅将系统状态分为故障倾向以及非故障倾向，所以该方法虽然能够准确的预测是否会出现故障，但故障发生的时间范围较难给定明确。

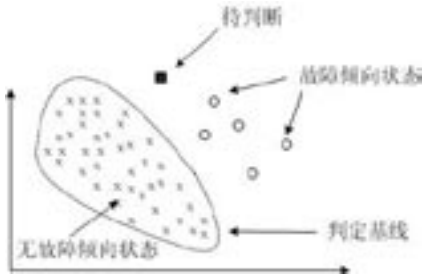

图5 基于分类的故障预测方法示意图

Hamerly与Elkan在[35]中提出一种名为NBEM的贝叶斯分类方法。该方法使用期望最大化方法对硬盘生产商昆腾公司提供的真实硬盘使用数据进行分析，而后通过混合模型为每个贝叶斯子模型m分配权重P(m)并通过期望最大化方法对该权重进行修正。最后按照边界条件对所有的子贝叶斯模型判定是否具有故障倾向。

Pizza等人[36]基于分类思想提出一种判定系统故障是否为永久性故障的方法。一旦系统的某组件的错误行为被发现（如，通过该组件的校验测试），该方法会找出该错误行为是由永久性故障还是瞬时故障造成的。虽然该文中没有提及故障预测工作，但是此类方法可以应用在故障预测方面。例如在网格环境中，用户在提交一个作业之前可以首先查看网格子环境的永久性故障数量以及瞬时故障数量，当故障数量超过某特定阈值时该环境即为具有故障倾向的环境，不再适宜提交作业。

Kiciman以及Fox等人[37]提出一种识别系统中故障组件的方法。他们通过构建决策树的方式对系统的运行路径进行分析，进而判定出系统中发生故障的组件。其中，系统运行路径指一系列系统运行轨迹变化，如组件升级，IP变更等，这些轨迹变化通过PinPoint[38]获得。经过一段时间的记录，系统获得大量运行路径，其中包括导向正常状态的运行路径以及导向异常乃至故障的运行路径，而后通过使用决策树的方式将两类运行路径进行分类。虽然该方法最开始被用于故障诊断，但其同样可以通过检测当前运行路径的变化情况对可能发生的故障进行预测。

Daidone等人[39]提出了基于隐式马尔可夫链（hidden Markov model, HMM）的方式判定所监控的系统组件当前是否真的处于健康状态。考虑到恶意故障的原因（如，拜占庭故障），系统监控值并不一定能够真实反映系统的真实状态，该方法将目标系统的历史行为特征作为输入，通过隐式马尔柯夫过程分析并预测系统行为特征及其走向，并与当前系统特征进行比较，从而获得系统状态是否与预期相符。

4.2.3基于系统模型的故障预测

如图6所示，此类预测方法根据被检查系统的无

故障行为建立系统模型，该模型的运算结果是系统正常运行时所期望得到的输出。将该模拟值与系统实际输出值通过比较器C进行比较，当二者不一致时，则说明系统运行已经不在预期范围内，可能出现异常乃至演化成失效。其中系统模型的输入可以是缓存于系统中的前一次的测量值，也可以是对系统组件实时的监测值。

通过记录目标系统的参数变化状况，分析测量值的变化趋势和范围是否与模拟的期望相符，进而判定系统是否进入异常状态。Elbaum等人[40]在采集邮件客户端的函数调用、组件更新以及配置参数变化时的系统特征的基础上，采取对比同操作的系统行为特征的方式，对系统的状态进行判定。该研究表明，系统参数出现单个的异常并不一定会导致故障，但当系统参数出现两组或两组以上连续的参数异常则极有可能导致出现故障。

当系统庞大而复杂时，使用随机模型(例如概率分布模型)往往能够更好的概括系统特征。Ward等人[41]通过对两个Web代理服务器间TCP通讯的均值和方差的分析，来判定Web服务是否存在性能故障。这种基于统计的方法的测量方法在[42]中有较为详尽的阐述，一天中某个时刻的测量值与系统之前的工作日（其中非工作日被单独处理）中相同的测量值进行比较，如果差异显著，则表明系统即将出现异常。

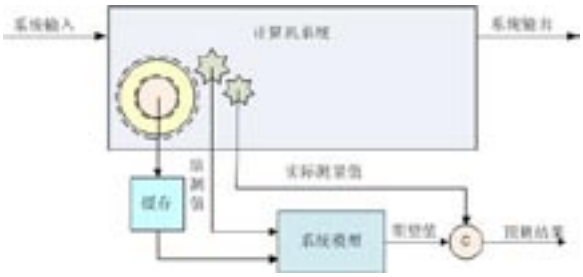

图6 基于系统模型的故障预测模型

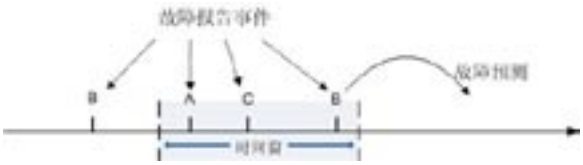

图7 基于事件驱动的故障预测方法示意图

图8 基于事件驱动的故障预测方法实现图

4.3 基于事件驱动的预测方法

如图7所示，当故障检测系统发现故障时往往会报告该故障并生成一个故障事件，因此基于事件驱动的预测的方法基于历史的故障报告事件集合进行分析。有别于之前的“基于状态的故障预测方法”，该方法不再是对系统输出的变量-时间曲线进行分析，而是将历史数据全部转化为事件，进行基于事件的分析以及预测。基于事件驱动的故障预测方法流程如图8所示，系统通过对历史故障事件进行分析，进行模式识别，该过程可以是通过人为实现或者通过程序识别，获得故障事件发生的时序关系，而后将该时序关系作为故障预测规则加入到预测系统中。

4.3.1 基于规则的故障预测方法

基于规则的故障预测方法直观而简单，利用系统或者人为添加预测规则，当系统符合预测规则时即表明即将产生故障。

Hatonen等人[43]是所能够查到的最早提出基于规则的故障预测方法，该方法通过分析故障日志获得响应的监控规则如“如果A事件以及B事件在五分钟之内相继发生，则C故障将在其后的10分钟内发生”，该方法还在监控规则中加入了诸多参数调整，如时间窗口范围、事件数量等。但该方法采取数据挖掘的方式由故障日志中生成规则，但是随后需要人工对机器生成的若干规则进行筛选，因此要求用户对目标系统有深入的了解。

Weiss等人[44]提出了一个被称为TimeWeaver的通用故障预测训练模型。不同于之前的人工添加规则或者直接从数据库搜索，该方法主要采用“抓取与扩充的方式”添加监控规则：算法开始时已经存在若干原始监控规则，这些监控规则可以是系统遗留或者人为添加的，该算法通过原始规则的匹配与混合生成新的监控规则。但新生成的监控规则显然并非全部有效，因此该方法对新生成的规则进行评估，结合预测的质量和多样性设定评估标准。经过筛选，得到若干新的规则后，此算法会将系统初始的规则剔除以形成新的规则集。该方法被用于网络与通信领域的设备故障预测，并与三种机器学习方

法RIPPER[45]，C4.5rules[46]，FOIL[47]进行了对比。

4.3.2 基于事件组合序列的故障预测

基于事件组合序列的故障预测方法试图从故障序列中查找到重复出现的或者关键的序列，并通过此类手段对故障进行预测。

Vilalta等人[48]利用色散帧技术(Dispersion Frame[49])对故障事件集合进行分析，得到故障事件的时间特征以及类型特征，并将分析结果存储到模式表中，当系统的行为与模式表中的模式相匹配时，则即可预测之后发生的故障。

Salfner等人在[50]提出了相似性事件预测(Similar Events Prediction)方法。该方法是一个半马尔柯夫链模型：每个故障事件对应到马尔科夫模型中的一个状态，两个相邻的故障之间的事件间隔被映射为连续时间状态的过渡期限，序列相似性的则由状态迁移可能性计算获得。系统的状态转移概率则是由模型通过对导致系统失效的错误事件序列和相关频率这两个维度的层次聚类来进行训练得到的。该方法是通过一个工业通讯系统的数据集进行验证的。

4.4 基于校验的故障预测方法

为了更主动地对系统故障进行预测，部分系统采用类似于校验和的方法进行故障的检测以及预测。例如，对于部分敏感文件系统，为了防止其被恶意篡改，可以通过数字签名，校验和等方式来验证其正确性。同理，故障发生直到系统失效的演变过程中会产生一系列的“差错症状”，通过检查类似于文件系统的正确性等方式可以有效预测到某些特定故障。该方法动态的校验系统中的一致性(如文件一致性，通信消息一致性等)。

Levy等人[51]发现Comverse语音邮件系统的子组件的故障频率是符合帕累托分布[52]的：大部分故障均由极少数的子系统产生，并且大部分的子系统不会产生任何故障(即Zipf分布[53])。该方法通过监控子系统的故障频率排序表，如果该表出现变化则预报即将出现故障。

5. 比较分析

本小节就以上所介绍的故障预测方法在模型、方法、适用场景等特点方面进行总结和比较，如表2所示。为了表述简洁，表2利用各预测方法所在的章节作为预测方法的分类编号。

表2 几种故障预测方法的比较

预测方法	类别	模型/方法	实验场景	适用范围
Csenki[20]	4.1.1	Jeilnski-Moranda模型	软件可靠性	① 软件质量控制
Pfefferman and Cernuschi-Frias[21]	4.1.1	非稳定伯努利模型	软件故障预测	① 减小系统动态变化带来的波动
Liang et al.[28]	4.1.2	故障数据基于时间及空间的压缩	BlueGene-L	② 长时间运行的应用
Fu and Xu[29]	4.1.2	同上	Wayne State大学网格计算环境	② 故障数量
Vaidyanathan et al.[57]	4.2.1	Semi-Markov reward 模型	UNIX系统下的内存消耗	③，④ 工作负载
Li et al.[30]	4.2.1	ARX模型	Apache web服务器的资源消耗	③，④ 系统稳定性控制
Hoffmann[33]	4.2.1	对系统调用的函数模拟	通信系统的性能故障	③，④ Apache web服务器的时间及内存的预测
Hamerly and Elkan[35]	4.2.2	NBEM贝叶斯分类方法	硬盘故障	基于独立假设对系统变量的监控
Pizza et al.[36]	4.2.2	贝叶斯最优分类	模拟组件的诊断	④ 区分系统的瞬时故障与永久故障
Kiciman and Fox[37]	4.2.2	构建决策树	J2EE应用服务器	基于运行时系统属性的预测
Daidone et al.[39]	4.2.2	HMM	模拟系统组件故障	探测系统不可靠因素
Elbaum et al.[40]	4.2.3	探测未知系统的函数调用等数据	采集邮件客户端“pine”的数据	单线程应用

预测方法	类别	模型/方法	实验场景	适用范围
Ward et al.[41]	4.2.3	利用随机模型对系统的某一特征进行描述	TCP代理服务器	③，④ 在稳定系统条件下，以但系统变量进行故障预测
Hatonen et al.[43]	4.3.1	机器学习生成规则 人工手动筛选规则	通信报警序列分析仪（TASA）	⑤ 需要对系统有充分的了解，不适用于动态系统
Weiss et al.[44]	4.3.1	Timeweaver自动生成规则	通信系统故障数据	⑤ 事件数据应包含若干属性，将某一时间段内的事件作为训练集
Vilalta et al.[48]	4.3.2	色散帧(Dispersion Frame)	系统故障消息数据	②，⑤
Salfner et al.[50]	4.3.2	相似性事件预测(Similar Events Prediction)方法	通信系统性能故障	⑤ 故障报告应包含故障类型和时间，处理事件序列
Levy et al.[51]	4.4	监控子系统的故障频率排序表	Comverse语音邮件系统	②，④ 预测系统组件故障，

① 基于故障的长期预测，② 面向分布式系统的故障预测，③ 面向运算资源的故障预测，
④ 面向系统服务的故障预测，⑤ 基于系统事件的故障预测

6. 未来研究趋势

故障预测技术已经经历了一段相当长的时期的发展。但是随着计算机技术的不断更新换代以及相关应用的不断丰富扩展，人们对它的发展也有着更为广泛而迫切的要求。结合目前的研究现状以及应用需求，我们认为其未来的研究趋势主要关注以下几个方面：

（1）面向新应用场景的故障预测方法的研究

性能良好的故障预测技术建立在深刻了解故障行为特点的基础之上。然而，具体的系统或应用因其自身的特点不同所呈现出的故障行为也不尽相同。随着大规模数据中心、云计算环境等新的应用场景的出现，新的问题和挑战也必将随之而来。具体应用所处环境的复杂性和高度动态性，以及在新环境中应用行为的变化都势必导致原有预测方法不能很好应用于新的场景模式。可见针对新的应用场景和应用模式研究出有效的预测方法是非常有必要的。

（2）基于预测结果的故障处理策略研究

在获得故障预测的结果后，结合具体应用场景，对故障避免以及宕机时间最小化等主动故障处理策略的研究是未来研究的关注方向。

（3）面向优化预测性能的数据分析方法研究

现有预测方法的依据都是来系统参数或数据的分析结果。因此分析方法和分析对象的选取不当就会导致预测方法性能上的下降。研究系统行为选取更为合适的系统参数分析或者结合数据特点对分析方法进行优化，通过对上述两方面的研究来达到优化预测方法性能的目的。

（4）预测系统的实际构建与部署研究

如何向具体的应用或系统中添加故障预测功能，即研究将定制的故障预测模块作为一种基础设施或功能方便植入服务对象。这是由技术方法向系统工具转化的过程。它涉及到系统的一致性、可扩展性等诸多问题，尤其是在面向网格计算、云计算等分布式计算环境时，系统的运行时监控、通信接口及数据持久化标准的设定等问题使得这一理论向实践转化的过程变得困难。可见，研究一种面向具体应用场景支持故障预测服务并且便于部署的预测系统框架是十分具有现实意义的。

7. 结束语

故障预测作为可靠性保障技术的一种重要手段，已经越来越得到研究者的重视。而且随着计算机体系结构的不断发展、运算规模的不断扩大以及服务模式不断更新，对故障预测的研究也必将随着模式和场景的变化而不断深入和拓展，可以说，故障预测的研究是一个十分具有现实和理论意义的领域。

本文回顾了近年来学术界对计算机系统故障预测领域的主要成果，对故障预测的相关概念、评估标准及技术方法进行了综述，并且依据分析对象的差异对故障预测提出一种分类方法，并在此基础上重点介绍了各种典型的故障预测方法，最后对各种预测方法进行了总结和对比，并对其未来的发展趋势进行了展望。

参考文献：

[1] Chakravorty, S., C.L. Mendes, and L.V. Kalé, Proactive fault tolerance in MPI applications via task migration[A], in High Performance Computing - HiPC 2006[C]. 2006, Springer. p. 485 - 496.

[2] Zheng, G., X. Ni, and L.V. Kalé. A scalable double in - memory checkpoint and restart scheme towards exascale[C]. in Dependable Systems and Networks Workshops (DSN - W), 2012 IEEE/IFIP 42nd International Conference on. 2012. IEEE.

[3] Lu, C. - D., Scalable diskless checkpointing for large parallel systems[D], 2005, University of Illinois.

[4] Bosilca, G., et al. MPICH - V: Toward a scalable fault tolerant MPI for volatile nodes[C]. in Supercomputing, ACM/IEEE 2002 Conference. 2002. IEEE.

[5] Engelmann, C. and G. Geist, A diskless checkpointing algorithm for super - scale architectures applied to the fast fourier transform[J]. Challenges of Large Applications in Distributed Environments, 2003.

[6] Ford, D., et al. Availability in globally distributed storage systems[C]. in Proceedings of the 9th USENIX conference on Operating systems design and implementation. 2010. USENIX Association.

[7] Dean, J. and S. Ghemawat, MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008. 51(1): p. 107 - 113.

[8] Melliar - Smith, P.M. and B. Randell. Software reliability: The role of programmed exception handling[C]. in ACM SIGOPS Operating Systems Review. 1977. ACM.

[9] Avizienis, A. and J. - C. Laprie, Dependable computing: From concepts to design diversity[J]. Proceedings of the IEEE, 1986. 74(5): p. 629 - 638.

[10] Laprie, J. - C. and K. Kanoun, Software reliability and system reliability[J]. Handbook for Software Reliability Engineering, 1996: p. 27 - 69.

[11] Siewiorek, D.P. and R.S. Swarz, Reliable computer systems: design and evaluation[M]. Vol. 3. 1998: AK Peters Massachusetts.

[12] Avizienis, A., et al., Basic concepts and taxonomy of dependable and secure computing[J]. Dependable and Secure Computing, IEEE Transactions on, 2004. 1(1): p. 11 - 33.

[13] Salfner, F., M. Lenk, and M. Malek, A survey of online failure prediction methods[J]. ACM Computing Surveys (CSUR), 2010. 42(3): p. 10.

[14] Altman, D.G., Practical statistics for medical research[M]. Vol. 12. 1991: Chapman & Hall/CRC.

[15] Weiss, G.M., Predicting telecommunication equipment failures from sequences of network alarms[J]. Handbook of Knowledge Discovery and Data Mining, 2002: p. 891 - 896.

[16] Van Rijsbergen, C.J., Information Retrieval, seconded[M]. Butterworth, London, U.K.Chapter 7.

[17] Blum, A. and T. Mitchell. Combining labeled and unlabeled data with co - training[C]. in Proceedings of the eleventh annual conference on Computational learning theory. 1998. ACM.

[18] Wang, Y. - M., et al. Towards dependable home networking: An experience report[C]. in Dependable Systems and Networks, 2000. DSN 2000. Proceedings International Conference on. 2000. IEEE.

[19] Gainaru, A., et al. Fault prediction under the microscope: A closer look into HPC systems[C]. in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. 2012. IEEE Computer Society Press.

[20] Aitchison, J. and I.R. Dunsmore, Statistical prediction analysis[M]. 1980: CUP Archive.

[21] Csenki, A., Bayes predictive analysis of a fundamental software reliability model[J]. Reliability, IEEE Transactions on, 1990. 39(2): p. 177 - 183.

[22] Melliar - Smith, P.M. and B. Randell. Software reliability: The role of programmed exception handling[C]. in ACM SIGOPS Operating Systems Review. 1977. ACM.

[23] Heiser, G., et al., Towards trustworthy computing systems: taking microkernels to the next level[J]. ACM SIGOPS Operating Systems Review, 2007. 41(4): p. 3 - 11.

[24] Brown, A. and D.A. Patterson. Embracing failure: A case for recovery - oriented computing (roc)[C]. in High Performance Transaction Processing Symposium. 2001.

[25] Babaoglu, O., et al., Self - star properties in complex information systems: conceptual and practical foundations[M]. Vol. 3460. 2005: Springer.

[26] Pfefferman, J.D. and B. Cernuschi - Fr í as, A nonparametric nonstationary procedure for failure prediction[J]. Reliability, IEEE Transactions on, 2002. 51(4): p. 434 - 442.
[27] Tang, D. and R.K. Iyer, Dependability measurement and modeling of a multicomputer system[J]. Computers, IEEE Transactions on, 1993. 42(1): p. 62 - 75.
[28] Liang, Y., et al. BlueGene/L failure analysis and prediction models[C]. in Dependable Systems and Networks, 2006. DSN 2006. International Conference on. 2006. IEEE.
[29] Fu, S. and C. - Z. Xu. Quantifying temporal and spatial correlation of failure events for proactive management[C]. in Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on. 2007. IEEE.
[30] Li, L., K. Vaidyanathan, and K.S. Trivedi. An approach for estimation of software aging in a web server[C]. in Empirical Software Engineering, 2002. Proceedings. 2002 International Symposium n. 2002. IEEE.
[31] Troudet, T. and W. Merrill. A real time neural net estimator of fatigue life[C]. in Neural Networks, 1990., 1990 IJCNN International Joint Conference on. 1990. IEEE.
[32] Wong, K., H. Ryan, and J. Tindle, Early warning fault detection using artificial intelligent methods[J]. 1996.
[33] Hoffmann, G.A., Failure Prediction in Complex Computer Systems[M]. 2006: Shaker Verlag, Aachen.
[34] Hoffmann, G. and M. Malek. Call availability prediction in a telecommunication system: A data driven empirical approach[C]. in Reliable Distributed Systems, 2006. SRDS ' 06. 25th IEEE Symposium on. 2006. IEEE.
[35] Hamerly, G. and C. Elkan. Bayesian approaches to failure prediction for disk drives[C]. in MACHINE LEARNING - INTERNATIONAL WORKSHOP THEN CONFERENCE - . 2001.
[36] Pizza, M., et al. Optimal discrimination between transient and permanent faults[C]. in High - Assurance Systems Engineering Symposium, 1998. Proceedings. Third IEEE International. 1998. IEEE.
[37] Kiciman, E. and A. Fox, Detecting application - level failures in component - based internet services[J]. Neural Networks, IEEE Transactions on, 2005. 16(5): p. 1027 - 1041.
[38] Chen, M.Y., et al. Pinpoint: Problem determination in large, dynamic internet services[C]. in Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on. 2002. IEEE.
[39] Daidone, A., et al. Hidden Markov models as a support for diagnosis: Formalization of the problem and synthesis of the solution[C]. in Reliable Distributed Systems, 2006. SRDS ' 06. 25th IEEE Symposium on. 2006. IEEE.
[40] Elbaum, S., S. Kanduri, and A. Amschler. Anomalies as precursors of field failures[C]. in Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on. 2003. IEEE.
[41] Ward, A., P. Glynn, and K. Richardson, Internet service performance failure detection[J]. ACM SIGMETRICS Performance Evaluation Review, 1998. 26(3): p. 38 - 43.
[42] Pettitt, A., Testing the normality of several independent samples using the Anderson - Darling statistic[J]. Applied Statistics, 1977: p. 156 - 161.
[43] Hatonen, K., et al. TASA: Telecommunication alarm sequence analyzer or how to enjoy faults in your network[C]. in Network Operations and Management Symposium, 1996., IEEE. 1996. IEEE.
[44] Weiss, G.M. Timeweaver: A genetic algorithm for identifying predictive patterns in sequences of events[C]. in Proceedings of the Genetic and Evolutionary Computation Conference. 1999.
[45] Cohen, W.W. Fast effective rule induction[C]. in MACHINE LEARNING - INTERNATIONAL WORKSHOP THEN CONFERENCE - . 1995. MORGAN KAUFMANN PUBLISHERS, INC.
[46] Quinlan, J.R., C4. 5: programs for machine learning[M]. Vol. 1. 1993: Morgan kaufmann.
[47] Quinlan, J.R., Learning logical definitions from relations[J]. Machine learning, 1990. 5(3): p. 239 - 266.
[48] Vilalta, R., et al., Predictive algorithms in the management of computer systems[J]. IBM Systems Journal, 2002. 41(3): p. 461 - 474.
[49] Lin, T. - T. and D.P. Siewiorek, Error log analysis: Statistical modeling and heuristic trend analysis[J]. Reliability, IEEE Transactions on, 1990. 39(4): p. 419 - 432.
[50] Salfner, F., Modeling event - driven time series with generalized hidden semi - Markov models[M]. 2006: Professoren des Inst. f ü r Informatik.
[51] Levy, D. and R. Chillarege. Early warning of failures through alarm analysis a case study in telecom voice mail systems[C]. in Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on. 2003. IEEE.

[52] Hosking, J.R. and J.R. Wallis, Parameter and quantile estimation for the generalized Pareto distribution[J]. Technometrics, 1987. 29(3): p. 339 - 349.
[53] Kingsley, Z.G., Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology[M]. 1949: Addison - Wesley Press.
[54] JELINSKI, Z. ANDMORANDA, Software reliability research[M]. In Statistical Computer Performance Evaluation, W. Freiberger, 1972: Ed. Academic Press.
[55] Neville, S.W. and B. Eng, Approaches for early fault detection in large scale engineering plants[J]. 1998.
[56] Castelli, V., et al., Proactive management of software aging[J]. IBM Journal of Research and Development, 2001. 45(2): p. 311 - 332.
[57] Vaidyanathan, K. and K.S. Trivedi. A measurement - based model for estimation of resource exhaustion in operational software systems[C]. in Software Reliability Engineering, 1999. Proceedings. 10th International Symposium on. 1999. IEEE.
[58] Guan, Q., Z. Zhang, and S. Fu, Ensemble of bayesian predictors and decision trees for proactive failure management in cloud computing systems[J]. Journal of Communications, 2012. 7(1): p. 52 - 61.
[59] Egwutuoha, I.P., et al. A Proactive Fault Tolerance Approach to High Performance Computing (HPC) in the Cloud[C]. in Cloud and Green Computing (CGC), 2012 Second International Conference on. 2012. IEEE.
[60] Samak, T., et al. Failure analysis of distributed scientific workflows executing in the cloud[C]. in Network and Service Management (CNSM), 2012 8th International Conference on. 2012. IEEE.
[61] Baldoni, R., et al., Online black - box failure prediction for mission critical distributed systems[A], in Computer Safety, Reliability, and Security[C]. 2012, Springer. p. 185 - 197.

要闻集锦

日本计划开发下一代超级计算机

据www.the - japan - news.com网站2013年5月9日消息报道，日本文部科学省通过了开发下一代超级计算机的计划，决定在2020年以前开发出下一代超级计算机，其性能运行速度将达到日本目前最快的超级计算机“京”（K Computer）的100倍，达到百亿亿次级。

“京”系统是世界上首台运算速度突破每秒1亿亿次的计算机，Linpack性能为10.51PFLOPS，不过在去年11月发布的TOP500排行榜中已落至第三位。超级计算机被认为是国家科技实力的指标，一些国家都在计划研发百亿亿次级超级计算机。日本文部科学省希望在2020年完成下一代超级计算机的开发，并准备在下一年度的预算概算要求中列入必要金额，制定详细计划。

下一代超级计算机将委托日本理化学研究所（RIKEN）担任研发，进一步与服务器制造商富士通合作，目前估计的投入研发预算将不超过“京”的开发费用1100亿日元，希望能抢先其他国家制造出百亿亿次级超级计算机，重新夺回世界超级计算机第一名的位置。

日本文部科学省官员表示，他们希望利用下一代超级计算机完成多项工作，比如开发新药，或是通过预测地震和海啸来预防灾害。

（肖 梅）