文章编号：2095-2163(2019)02-0215-06 中图分类号：TP338.8 文献标志码：A

分布式系统中失效检测器综述

刘家希，吴智博，董 剑，温东新
（哈尔滨工业大学 计算机科学与技术学院，哈尔滨 150001）

摘 要：失效检测器是构建高可用分布式系统的基础组件之一，能够保证分布式系统提供持续、可靠的服务，以最低的检测负载实现快速、准确的失效检测为目标。目前的失效检测器主要围绕自适应失效检测和检测结果共享机制展开研究，以期能够在检测时间、检测准确性以及检测负载等失效检测服务质量方面不断改进。

关键词：分布式系统；失效检测器；服务质量

Summarization on failure detector in distributed system

LIU Jiaxi, WU Zhibo, DONG Jian, WEN Dongxin
(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

【Abstract】 Failure detector is one of fundamental components to build high availability distributed systems, and can ensure that the distributed systems provide the continuous and reliable service. The target of failure detector is to achieve the fast and accurate failure detection with the lowest overhead. At present, in order to improve the detection time, accuracy and overhead, the failure detector mainly focuses on adaptive failure detection and mechanism of sharing result.

【Key words】 distributed system; failure detector; Quality of Service

0 引 言

分布式系统(Distributed System)是一个硬件或者软件分布在不同的网络计算机上，彼此之间仅仅通过消息传递进行通信和协调的系统[1]。目前，分布式系统可以说是无所不在，无论商业、学界、政府部门或者是家庭之中均能看到其身影。这些分布式系统提供共享资源、共享数据的技术手段，在这个信息化的时代是极其重要的[2]。然而，随着分布式系统规模的不断增长以及复杂性的不断增加，由于各种原因造成的软硬件失效已然不可避免，这就在相当程度上决定了分布式系统可用性的好坏。容错技术是分布式系统中保证高可用性的有效方法[3]，可以使系统在有部分硬件或者软件发生失效的情况下，仍可作为整体能够正常运行并完成其设计功能。而失效检测器是容错技术实现的前提，通过周期性地探测系统中节点的状态，可以为系统容错提供良好的信息支持。因此，失效检测器的性能直接影响分布式系统的正常运行。

目前，失效检测器的研究主要集中在失效检测器理论模型与失效检测算法的实现两个方面。总地来说，在失效检测器理论模型方面，Chandra 等人[4]首次提出了失效检测器的理论，并且根据完整性与准确性定义了 8 种类型的失效检测器，同时指出了解决分布式一致性问题最弱的失效检测器模型。在此基础上，陆续有学者针对不同的分布式系统模型提出不同能力的失效检测器模型。而在失效检测器的实现方面，针对失效检测所面临的检测准确性、速度以及负载的矛盾，自适应失效检测器和基于结果共享机制的失效检测器也屡获提出，并相继问世。这些失效检测器以不同的侧重点解决了实际分布式系统中对失效检测服务的需求设计问题。

本文以失效检测器的实现为重点分析对象，并将从如下方面展开论述：首先，提出失效检测器定义、实现以及服务质量评价指标。其次，研究了不同种类的失效检测器。最后，对分布式系统中失效检测器做出总结与展望。

1 失效检测器模型

1.1 失效检测器定义

Chandra 等人[4]最早提出了失效检测器的理论。在该理论中，分布式系统中每个节点都有一个本地的

基金项目：国家自然科学基金(61100029,61370085)。

作者简介：刘家希(1988-)，男，博士研究生，主要研究方向：容错计算、分布式计算；吴智博(1954-)，男，博士，教授，主要研究方向：容错计算、普适计算、移动计算等。

收稿日期：2018-05-26

失效检测器,用于检测系统中的部分节点并且维护一张怀疑节点发生失效的列表。失效检测器可能会错误地将正常运行的节点加入到怀疑列表中,但当发现错误怀疑时,又可以把该节点从怀疑列表中直接删除,这个过程可以重复进行。任意 2 个节点的失效检测器对相同的节点可能有不同的检测结果。

失效检测器形式化的定义可表述为:在一个由 n 个节点组成的分布式系统 Π 中,F 被称为失效模式,$F:T\rightarrow 2^{\Pi}$。其中,T 是全局时钟到自然计数的一个映射。对于 Π 中的节点 p,则用 $crash(F)=\bigcup_{t\in T}F(t)$ 定义发生失效的节点集合,$correct(F)=\Pi-F(t)$ 定义被判断为正确的节点集合。

一个失效检测器(Failure Detector,FD)可视作为一组失效检测模块的集合,分布在系统的每一个节点上,而失效检测器的输出就是被怀疑发生失效的节点的集合,可以用 H 来表示,$H:\Pi\times T\rightarrow 2^{\Pi}$,$H(p,t)$ 则表示节点 p 在 t 时刻所认为发生失效的节点集合。

1.2 失效检测器的实现

当前失效检测方法的实现大多数都是基于心跳机制的。通过判断被检测节点发出的消息能否在规定时间内到达,从而判定被检测节点是否发生失效。按照实现方式的不同,可以分为 PUSH 模型[5] 和 PULL 模型[1],其它的失效检测方法都是建立在这 2 种失效检测方法之上。现对这 2 种方法将给出如下阐释与分析。

(1)在基于 PUSH 模型的失效检测方法中,被检测节点是主动的参与者,检测方法的基础工作方式如图 1 所示。由图 1 看到,被检测节点 p 周期性地发送“I am alive”消息到失效检测器,这种消息也称为心跳消息。失效检测器会设置一个超时值 Δt,如果失效检测器在超时值 Δt 前没有收到被检测节点发送的消息,则怀疑被检测节点发生失效。由于 PUSH 模型在系统内是单方向的消息传输,使其可获得较高传输效率。同时如果被检测节点有多个检测者,可以使用硬件的多播机制来实现。但是,PUSH 模型需要被检测节点维护本地时钟以便周期性地发送消息。

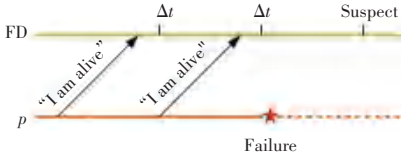

图 1 PUSH 模型

Fig. 1 PUSH model

(2)在基于 PULL 模型的失效检测方法中,被检测节点是被动的参与者,检测方法的基础工作方式如图 2 所示。失效检测器周期性地发送“Are you alive”查询消息到被检测节点,并且设置一个超时值 Δt。如果被检测节点在超时值 Δt 前回复应答消息“Yes”,则失效检测器判定其处于正常状态;否则,失效检测器怀疑其发生失效。由于被检测节点在 PULL 模型中是被动参与者,所以失效检测器更容易实现。而且,这些被检测节点只需要在有询问消息时做出反应,不需要为了定期发送消息而维护当地时钟,可以在异步计算环境下运行。但是,采用 PULL 模型的失效检测方法所需检测消息是 PUSH 模型的 2 倍。

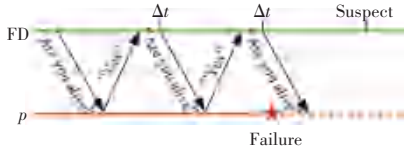

图 2 PULL 模型

Fig. 2 PULL model

1.3 失效检测器的服务质量

许多实际应用对失效检测器所提供的节点怀疑信息有诸多限制,而且也无法接受过慢的或者输出错误太多的失效检测服务。针对这个问题,Chen 等人[6] 在失效检测器理论模型的基础上,提出了一套失效检测器服务质量(Quality of Service,QoS)的评价指标体系,可以对失效检测器性能进行定量评价。这套指标体系主要描述失效检测器发现真实失效的速度以及避免错误怀疑的概率两个方面。

为了理解 Chen 所创建的定量评价指标体系,需要明晰一些 QoS 关键指标含义。其中,T 表示失效检测器认为被检测节点处于正常状态,S 表示失效检测器认为被检测节点处于失效状态。$T-transition$ 表示失效检测器的输出结果由 S 变成 T,$S-transition$ 表示失效检测器的输出结果由 T 变成 S。详情表述见如下。

(1)检测时间 T_D:从节点发生失效的时刻到该节点被失效检测器永久怀疑所持续的时间。也可以说,T_D 表示节点发生失效的时刻到最后一个 $S-transition$ 发生的时刻(以后不会再发生 $transition$)之间持续的时间,如图 3 所示。

(2) 错误间隔时间 T_{MR}:失效检测器连续 2 次发生错误怀疑的间隔时间。也就是说,T_{MR} 表示从一个 $S-transition$ 到下一个 $S-transition$ 所经历的时间,如图 3 所示。

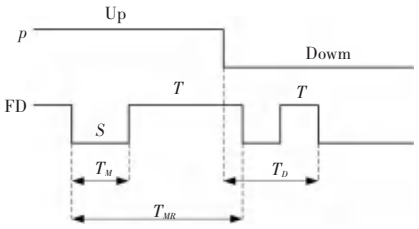

图 3　QoS 指标

Fig. 3　QoS metrics

（3）错误持续时间 T_M：失效检测器改正一次错误怀疑所经历的时间。也就是说，T_M 表示从一个 $S-transition$ 到下一个 $T-transition$ 所经历的时间，如图 3 所示。

2　失效检测器的分类

失效检测器不仅需要保证基本的失效检测能力，也需要能够保证失效检测能力在可接受的服务质量范围内。随着大规模分布式系统的发展，就需要检测大量的节点，较高概率的消息丢失、不断变换的系统拓扑以及消息延迟的不可预测性等问题不断涌现。这些情况对于研究开发高效的失效检测器提出了巨大的挑战，加剧了失效检测有效性（失效检测准确性和失效检测速度）和失效检测所需负载的矛盾。而自适应失效检测与结果共享机制是当前解决这一问题的主要途径。因而围绕这 2 个方面，针对当前大规模分布式系统失效检测技术的研究可做综合分析与详述如下。

2.1　自适应于网络环境的失效检测器

在早期的自适应失效检测器的研究中，通过调整心跳消息发送间隔 η 和超时值 Δt 来适应网络环境的变化，从而在一定检测速度的约束下，尽力提高检测准确性。例如，Sotoma 等人[7]提出了一种基于 MEAN 超时值预测方法的自适应失效检测器，并且在 CORBA 系统中实现了这种自适应失效检测器。这种失效检测器使用心跳传输时间的平均值作为超时值 Δt 的预测值，可以周期性地调整超时值 Δt，从而达到了自适应失效检测设计效果。Fetzer 等人[8]提出了另一种基于 MAX 超时值预测方法的自适应失效检测协议，通过利用历史心跳消息的传输时间最大值作为超时值 Δt 的预测值，可以有效地改善失效检测准确性。Falai 等人[9]对这些超时值预测方法进行了总结与归纳，选取出 5 种超时值预测方法：LAST、MEAN、WINMEAN(N)、LPF(β) 以及 ARIMA(p,d,q)，分别为其配以不同的安全边界（SM_{CI} 和 SM_{JAC}），并通过实验的方法对其性能进行评价。实验结果表明，ARIMA(p,d,q) 方法具有较高的预测准确性，而具有较低算法复杂性的 LAST 方法结合 SM_{JAC} 安全边界则获得了最快的检测时间以及较好的检测准确性。

上述这些方法对 QoS 的研究都停留在定性的阶段，并不能够精确地描述失效检测器所能满足的 QoS，而且用户也不能对失效检测器提出定量的 QoS 需求。针对这一情况，Chen 等人[6]提出了一整套失效检测器的 QoS 评价指标体系。这套 QoS 指标体系主要是对失效检测器的检测速度以及准确性的衡量，实现了对失效检测器的检测能力的定量的评价。同时，基于以上 QoS 指标，Chen 等人提出的自适应失效检测器 NFD-E 可以根据用户或者应用给定的 QoS 需求，自动地调整检测参数，从而以最小的检测负载获得定量的 QoS 指标。针对 NFD-E 检测器检测速度的问题，Bertier 等人[10]提出了一种基于动态安全边界的失效检测器。这种失效检测器利用 Jacobson 的往返时间（round-trip time，RTT）来对安全边界进行估计，使其可以随网络环境的变化进行动态改变，此方法明显地降低了 Δt 值，提高了检测速度。但是，这种改进后的 NFD-E 检测器在检测速度提高的同时带来了更多的错误检测。针对 NFD-E 检测器检测准确性的问题，Tomsic 等人[11]提出了一种基于双窗口的失效检测器 2W-FD。这种失效检测器使用 2 个不同尺寸的滑动窗口存储历史心跳消息，分别计算超时值，从结果中选取较大的值作为最终的超时值，这种方法可以捕获更多的心跳消息，从而提高检测准确性。实验结果表明，在不稳定的网络环境下，2W-FD 检测器能够获得更高的检测准确性。

在大规模分布式系统中，尤其是部署大量不同类型分布式应用的系统，失效检测器仅提供单一的 QoS 已经不能满足应用的需求。Défago 等人[12]提出的新的失效检测器模型—Accrual 检测器，可以同时满足大规模分布式系统中多个应用不同的 QoS 需求。该次研究将传统失效检测器模型中的监测权与解释权相分离。Accrual 检测器只负责对节点进行监测，与此同时还将输出一个代表被检测节点怀疑程度的连续值（也称为怀疑值，$sl_{qp}(t)$），而不再直接输出相信或者怀疑的二值性检测结果。不同的应用根据自身 QoS 需求对此怀疑值做出解释，从而决定被检测节点的状态。

φ 检测器[13]是 Accrual 检测器的首款设计，如图 4 所示。通过设置滑动窗口存储来自被检测节点的心跳消息，然后假设这些心跳消息的到达时间间

隔符合正态分布，从而根据累积分布函数 $F(t)$ 计算怀疑值 φ。应用程序可以在任意时刻 t_{now} 发起查询，会收到检测器返回一个 φ 值，最终通过 φ 值与应用程序设定的阈值的比较结果确定被检测节点状态。φ 检测器的实现满足了应用程序对失效检测服务的多样化需求。

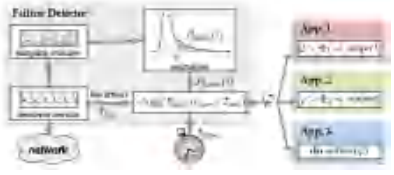

图 4 φ 检测器

Fig. 4 φ failure detector

Xiong 等人[14]指出指数分布更加适合于描述心跳消息到达时间间隔，从而利用指数分布计算怀疑值。Lakshman 等人[15]将这种基于指数分布的 Accrual 检测器应用于著名的社交网站 Facebook 的 P2P 存储系统 Cassandra 中，获得了良好的失效检测性能。

同时，也陆续推出了一些基于其它方法处理心跳消息的 Accrual 检测器的研发实例。Satzger 等人[16]提出了一种低开销的 Accrual 检测器的设计，利用直方统计图计算怀疑值，通过直接分析检测消息延迟的历史记录，计算延迟小于当前检测时间的消息所占的比例，以此为基础计算怀疑值。He 等人[17]提出的 Accrual 检测器则引入了指数移动平均算法，利用心跳消息到达时间间隔与间隔预测值计算怀疑值。

2.2 检测结果共享机制

在节点检测层面，自适应失效检测可以解决检测有效性与检测负载之间的关系问题。然而从整个系统层面来看，如果每个节点都单独进行失效检测，那么在一个规模为 n 的系统中，至多需要产生 n^2 个检测关系。这对于大规模的分布式系统而言，无疑会造成巨大的检测开销。因此，节点之间通过共享检测结果的方法降低检测开销，以期改善检测性能获得了广泛的研究。在众多方法中，基于层次式的检测方法和基于 Gossip 式的检测方法已成为学界瞩目焦点。对此内容详情可做研究评述如下。

(1)层次式方法。将系统中节点组织成树或森林的层次结构，然后依托这些特殊的结构建立失效检测关系，以达到降低系统检测负载，提高可扩展性的目的。一种典型的层次式结构即如图 5 所示。

Felber 等人[18]提出了一种层次式的失效检测服务。在系统中，节点根据 IP 地址被划分到不同的子网中，每一个子网中的节点为一个分组，分组中选择一个主节点部署失效检测模块，负责对分组中其它节点进行检测。不同的分组之间可以通过主节点进行检测结果交换。这种方法易于管理，并且可以迅捷实现失效检测。但是，该方法却高度依赖系统拓扑结构，并不适合拓扑频繁改变的系统。

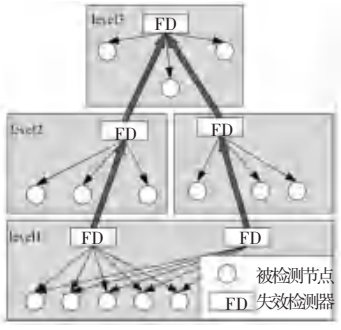

图 5 层次式失效检测

Fig. 5 Hierarchical failure detector

Stelling 等人[19]根据网格的架构特点，利用 Globus toolkit 工具包研发了一个类两层结构的检测协议。每台主机上安装一个本地检测模块，负责检测主机上运行的所有进程，不同主机上的检测模块会互相交换信息以获得检测结果的全面共享。这一协议适用于节点计算能力较强的系统，而且从节点层面看，这不是一种真正意义上的双层检测结构。而 Bertier 等人[20]提出了一种真正意义上的双层架构的检测协议。系统中所有节点被分配到不同分组中，每一个分组选择一个主节点部署失效检测器，负责检测组内的节点状态，而所有的主节点通过互相交换检测结果可形成全局检测信息。这种检测架构在整体设计上相对容易且效率较高，但是与多层结构相比，其对检测负载的改进没有多层结构明显。

(2) Gossip[21]。本身是一种概率多播协议[22]，可采用类似蠕虫病毒(rumor)的快速传播方式，通过多轮的消息交换以较低的代价实现消息的高效广播。Renesse 等人[23]将 Gossip 这种协议引入到失效检测领域中，提出了 2 种基于 Gossip 式的失效检测器：基本 Gossip 检测器以及多层 Gossip 检测器。对于基本 Gossip 检测器而言，每轮检测中节点随机选择邻居节点进行检测并交换检测结果，通过数轮的交换可以获得系统中其它节点的状态。在这种检测器中，其检测负载与系统的规模以及拓扑结构无关，而其检测时间受算法随机性的影响，随着系统规模的增大，检测时间会呈现线性增长。针对这种情况，多层 Gossip 检测器也随即得以提出。借鉴层次式的检测思想，根据节点 IP 信息将系统中节点划分到

不同的子网，大多数的 Gossip 消息在子网内传播，只有少数的消息可以跨越子网传播，实现全局的检测需求。在这种检测器中，其检测负载只和系统子网的数量相关，而与具体节点数量无关，拥有良好的可扩展性。然而，针对某一确定节点的检测时间至少需要 $O(logn)T_{fail}$。其中，T_{fail} 是对某一节点所允许的失效检测延迟。

虽然基于 Gossip 式检测方法在降低检测负载以及提高可扩展性方面具有优势，但是其所需检测时间仍然较长。为此，大量的研究集中在对检测时间的改进上。例如，Gupta 等人[24]利用“再检测”机制实现的组成员协议 SWIM。在 SWIM 协议中，节点 q 随机选择一个节点 p 进行检测，如果在规定时间内没有收到回复消息，那么节点 q 会随机选取 k 个节点再对节点 p 进行检测，最后根据这 k 个节点返回的检测结果对节点 p 的状态做出判断，以上设计过程如图 6 所示。这种方法可以有效地提高检测准确性，并且通过调整检测周期与协议参数改进检测时间。Snyder 等人[25]将 SWIM 协议应用到大规模 HPC 存储系统，通过选取适当的协议参数以及检测周期，可以获得良好的检测性能。Horita 等人[26]提出了类似方法，同样需要另外 k 个节点进行辅助检测，只是这种辅助检测是静态配置的，不需要动态发起过程，在提高检测准确性的同时进一步改善了检测时间。Ward 等人[27]针对大规模云计算系统提出了一种多层的 gossip 协议。根据云计算系统架构，可将系统分为 3 层：云、分组以及虚拟机，在每个检测周期内，虚拟机选择同分组内其它虚拟机进行消息交换，其中一定比例的虚拟机会跨越分组、以及跨越云进行消息交换。这种方法可以利用实际系统不同层次所具有的不同通信延迟，实现快速的失效检测。

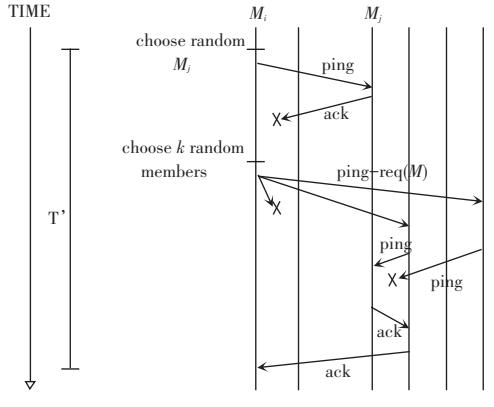

图 6　SWIM 失效检测器检测过程

Fig. 6　Process of failure detector in SWIM

3　结束语

失效检测器已经是构建高可用分布式系统的基础组件之一。失效检测器通过周期性地探测系统中节点的状态，不仅能够解决分布式系统中一些基础问题（例如，一致性问题，原子广播等），还能够为路由选择、负载均衡以及任务调度等发挥有益的支持作用。因此，失效检测器所提供的服务对分布式系统的性能有着重要的影响。自适应失效检测和检测结果共享机制是分布式系统中失效检测实现的主要方法，未来将深入研究如何降低检测负载，提高检测准确性以及检测时间。

参考文献

[1] 常光辉. 大规模分布式可信监控系统研究[D]. 重庆：重庆大学，2011.

[2] 张家琳. 分布式计算中的共识问题研究[D]. 北京：清华大学，2010.

[3] 李磊. 分布式系统中容错机制性能优化技术研究[D]. 长沙：国防科学技术大学，2007.

[4] CHANDRA T D，TOUEG S. Unreliable failure detectors for reliable distributed systems[J]. Journal of the ACM (JACM). 1996，43(2)：225-267.

[5] LARREA M，ANTA F A，ARÉVALO S. Optimal implementation of the weakest failure detector for solving consensus[C]// The 19th IEEE Symposium on Reliable Distributed Systems (SRDS). Nürnberg，Germany：IEEE Computer Society Press，2000：52-59.

[6] CHEN Wei，TOUEG S，AGUILERA M K. On the quality of service of failure detectors[J]. Computers，IEEE Transactions on，2002，51(1)：13-32.

[7] SOTOMA I，MADEIRA E R M. Adaptation-algorithms to adaptive fault monitoring and their implementation on Corba[C]// The 3rd International Symposium on Distributed Objects and Applications. Rome，Italy：IEEE，2001：219-228.

[8] FETZER C，RAYNAL M，TRONEL F. An adaptive failure detection protocol[C]// Pacific Rim International Symposium on Dependable Computing，Seoul，Korea：IEEE，2001：146-153.

[9] FALAI L，BONDAVALLI A. Experimental evaluation of the Qos of failure detectors on wide area network[C]//Proceedings of International Conference on Dependable Systems and Networks. Yokohama，Japan：IEEE Press，2005：624-633.

[10] BERTIER M，MARIN O，SENS P. Implementation and performance evaluation of an adaptable failure detector[C]// DSN ′02：Proceedings of the 2002 International Conference on Dependable Systems and Networks. Washington，DC，USA：IEEE Computer Society，2002：354-363.

[11] TOMSIC A，SENS P，GARCIA J，et al. 2W-FD：A failure detector algorithm with QoS[C]//International Parallel and Distributed Processing Symposium (IPDPS2015). Hyderabad，India：IEEE Press，2015：885-893.

[12] DÉFAGO X，URBAN P，HAYASHIBARA N et al. Definition and specification of accrual failure detectors[C]//International

Conference on Dependable Systems and Networks. Yokohama, Japan: IEEE Computer Society, 2005: 206-215.
[13] HAYASHIBARA N, DÉFAGO X, YARED R, et al. The Φ accrual failure detector [C]//Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems. Florianopolis, Brazil: IEEE Press, 2004: 66-78.
[14] XIONG N, DÉFAGO X. ED FD: Improving the Phi accrual failure detecor[R]. Japan: JAIST, 2007.
[15] LAKSHMAN A, MALIK P. Cassandra: A decentralized structured storage system[J]. ACM SIGOPS Operating Systems Review, 2010, 44(2): 35-40.
[16] SATZGER B, PIETZOWSKI A, TRUMLER W, et al. A new adaptive accrual failure detector for dependable distributed systems [C]// Proceedings of ACM symposium on Applied computing (SAC '07). Seoul, Korea: ACM Press, 2007: 551-555.
[17] HE Yanzhang, JIANG Xiaohong, DAI Changbo, et al. Self-adaptive failure detector for peer-to-peer distributed system considering the link faults[M]//DOU Y, LIN H, SUN G, et al. Advanced parallel processing technologies. APPT 2017. Lecture Notes in Computer Science, Cham: Springer, 2017, 10561: 64-75.
[18] FELBER P, DÉFAGO X, GUERRAOUI R, et al. Failure detectors as first class objects [C]// Proceedings of the International Symposium on Distributed Objects and Applications. Edinburgh, United Kingdom: IEEE Computer Society, 1999: 132-141.
[19] STELLING P, FOSTER I, KESSELMAN C, et al. A fault detection service for wide area distributed computations [C]// Proceedings of The Seventh International Symposium on High Performance Distributed Computing. Chicago, USA: IEEE Computer Society, 1998: 268-278.
[20] BERTIER M, MARIN O, SENS P. Performance analysis of a hierarchical failure detector [C]// Proceedings of 2003 International Conference on Dependable Systems and Networks. San Francisco, CA, USA: IEEE Computer Society Press, 2003: 635-644.
[21] LIN Mengjiang, MARZULLO K, MASINI S. Gossip versus deterministic flooding: Low message overhead and high reliability for broadcasting on small networks[R]. CA, USA: University of California, 1999.
[22] EUGSTER P T, GUERRAOUI R. Probabilistic multicast[C]// Proceedings of International Conference on Dependable Systems and Networks. Washington, DC, USA: IEEE Computer Society, 2002: 313-322.
[23] VAN RENESSE R, MINSKY Y, HAYDEN M. A gossip-style failure detection service[C]// Proceedings of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing Middleware. The Lake District, UK: ACM, 1998: 55-70.
[24] GUPTA I, CHANDRA T D, GOLDSZMIDT A S. On scalable and efficient distributed failure detectors [C]//Twentieth ACM Symposium on Principles of Distributed Computing. Newport, Rhode Island: ACM Press, 2001: 170-179.
[25] SNYDER S, CARNS P, JENKINS J, et al. A case for epidemic fault detection and group membership in HPC storage systems [M]//JARVIS S, WRIGHT S, HAMMOND S. High performance computing systems. performance modeling, Benchmarking, and Simulation. PMBS 2014. Lecture Notes in Computer Science, Cham: Springer, 2014, 8966: 237-248.
[26] HORITA Y, TAURA K, CHIKAYMA T. A scalable and efficient self-organizing failure detector for grid applications[C]//The 6th IEEE/ACM International Workshop on Grid Computing. Seattle, WA, USA: IEEE Computer Society Press, 2005: 202-210.
[27] WARD J S, BARKER A. Monitoring large-scale cloud systems with layered gossip protocols[J]. arXiv preprint arXiv: 1305.7403, 2013.

(上接第214页)

实验表明,改进后的程序胜率很大,证实了本程序改进的有效性。对 BistuHex2 程序算法的修改能够提高算法搜索精确度,在一定程度上增加了找到最优落子坐标的概率。通过加大 α-β 剪枝算法搜索的深度,以及更改 UCT 搜索可走棋子的个数可以让搜索更加准确,从而大幅提升了算法效率。

5 结束语

通过对之前仅用 UCT 算法的程序进行了改进,根据 α-β 剪枝算法的特点与 UCT 和电阻电路评估策略合理结合,很好地提高了最优点的搜索效率。但是本程序并没有体现深度学习的算法及思路,在今后的工作中,研究将会继续丰富界面程序功能,添加更为人性化的交互界面,学习参考 Alpha Go 和 Alpha Zero 思想与方法,旨在进一步完善海克斯棋程序。

参考文献

[1] Baidu 百科. 六贯棋[EB/OL]. [2018]. https://baike.baidu.com/item/%E5%85%AD%E8%B4%AF%E6%A3%8B/10095237?fr=aladdin.
[2] 中国人工智能学会机器博弈专业委员会. 中国大学生计算机博弈大赛棋谱标准说明书[Z]. 沈阳:中国人工智能学会机器博弈专业委员会,2018.
[3] ANSHELEVICH V V. The game of Hex: An automatic theorem-proving approach to game programming[C]// Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence. Austin, TX: AAAI Press, 2000: 189-194.