引用格式:周星宇,李春涛,姚瑞,等. 无人机飞行控制软件负荷均衡策略设计[J]. 电光与控制,2018,25(6):25-30. ZHOU X Y, LI C T, YAO R, et al. Design of load balancing strategies for UAV flight control software[J]. Electronics Optics & Control, 2018, 25(6):25-30.

无人机飞行控制软件负荷均衡策略设计

周星宇, 李春涛, 姚 瑞, 范 影
(南京航空航天大学自动化学院,南京 210016)

摘 要: 针对无人机飞行控制软件功能复杂、实时性要求高等特点,以基于CAN总线的分布式架构的飞行控制计算机为平台,利用软件总线设计了飞行控制软件的静态和动态负荷均衡策略,并对解算节点之间功能模块的迁移方法进行了设计,解决了分布式架构的飞行控制系统中由于节点负载不均衡导致系统运行效率下降和资源浪费的问题,保证了整个系统的持续高效运行。最后进行了实验验证,测试结果表明,两种负荷均衡算法均能准确地进行功能模块调度,具有良好的实时性,提高了分布式架构无人机飞行控制系统的可靠性和安全性。

关键词: 无人机; 飞行控制软件; 软件总线; 分布式架构; 负荷均衡
中图分类号: V279; TP273.5 **文献标志码**: A **doi**:10.3969/j.issn.1671-637X.2018.06.006

Design of Load Balancing Strategies for UAV Flight Control Software

ZHOU Xing-yu, LI Chun-tao, YAO Rui, FAN Ying
(College of Automation, Nanjing University of Aeronautics & Astronautics, Nanjing 210016, China)

Abstract: Aiming at the UAV flight control software with complicated functions and high real-time requirements, and taking the flight control computer with distributed architecture based on CAN bus as a platform, we designed the strategies for static and dynamic load balancing by utilizing software bus. In addition, the mechanism of function module migration between the nodes was designed. These methods could solve the problems caused by the load imbalance of nodes in the distributed flight control system, i.e., declining of operating efficiency and wasting of resources, and guarantee the efficient and continuous operation of the system. Finally, experimental verification was made, and the result showed that: both types of load-balancing algorithms can schedule the functional modules accurately and guarantee the real-time performance, and thus the reliability and security of the distributed flight control system are improved.

Key words: UAV; flight control software; software bus; distributed architecture; load balancing

0 引言

随着无人机飞行任务越来越多样化,功能越来越复杂,飞行控制系统的任务负载日益繁重[1],为提高飞行控制系统性能,出现了分布式架构的飞行控制系统,系统各节点均进行任务解算,需要将解算任务合理分配至多个节点上,共同完成复杂的飞行控制任务[2]。与此同时,随着系统的运行,各个功能模块的负载会发生变化,进而导致各个节点的负荷变得不均衡,造成系统运行效率下降以及资源的极大浪费。

针对上述问题,本文以基于CAN总线的飞行控制计算机为硬件平台,VxWorks操作系统为软件平台,利用软件总线设计了飞行控制软件的静态和动态负荷均衡策略,对各个解算节点的负荷进行合理分配,保证了各个节点的高效持续运行,进而提高了飞行控制系统的可靠性。

收稿日期:2017-07-15 **修回日期**:2017-08-08
作者简介:周星宇(1993—),男,江苏盐城人,硕士,研究方向为无人机飞行控制系统与嵌入式软件开发。

1 飞行控制软件运行平台

1.1 硬件平台

基于CAN总线的分布式架构飞行控制计算机如图1所示。该目标机由4个节点组成,节点之间采用分工协作的方式,分别负责不同的功能模块,通过CAN总线进行数据的通信。与其他总线相比,CAN总线数据传输速度较高、具有自我诊断能力以及抗干扰能力强的优势[3],满足了飞行控制系统内部通信的需求。

系统的功能和资源分布在不同的节点上，当某一个节点负荷不均衡时，可执行相应的处置逻辑，将部分功能模块进行迁移，达到分担负荷的目的。同时，当系统接口需求增加时，可通过增加节点实现接口资源的扩展。

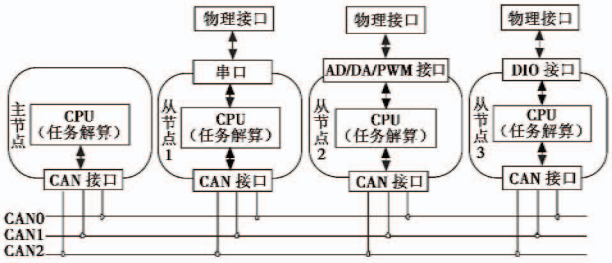

图1 飞行控制计算机结构

Fig. 1 The structure of flight control computer

1.2 软件平台

飞行控制系统作为硬实时系统，对系统的响应时间有严格的要求，要求系统对外部事件能够做到快速响应以及多任务处理。VxWorks 是专门为实时嵌入式系统设计开发的操作系统软件，具有易裁剪、高性能、高可靠性以及高实时性等优点[4]。VxWorks 高效的实时任务调度、较短的中断调度时间使其能够满足飞行控制软件对实时性的要求。

2 软件总线结构

为了能够实现整个系统的负荷分担和功能模块迁移，需要软件总线提供功能模块的注册和删除、资源的统一管理和调度以及保证各节点内存资源一致性的功能。软件总线的结构如图2所示。飞行控制软件按功能和任务间的耦合性大小划分为遥控遥测功能模块、传感器数据采集模块、传感器数据融合模块、故障监测与处置模块、导航制导模块、控制律解算模块及控制输出管理模块等，对其进行统一的接口封装使其能够方便地挂接到软件总线上。软件总线内部的模块管理器可以对模块进行相应的加载和删除[5]。同时，软件总线能够利用自身的资源管理器对内存资源池进行管理，完成节点内不同功能模块之间数据资源的交互以及节点之间数据的传输，保证节点的解算同步[6-7]。

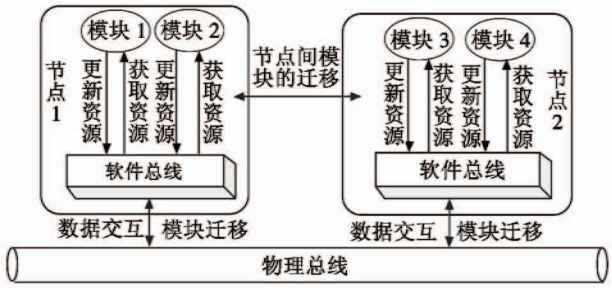

图2 软件总线结构图

Fig. 2 The construction of software bus

3 节点间的负荷分担策略设计

3.1 功能模块运行时间与负载统计

飞行控制系统节点的负载与节点中运行的功能模块数量、模块的运行时间和调用频率有关，即模块数量越多，运行时间越长，调用频率越高，则 CPU 的利用率越高。因此，定义模块的负载为1 s内模块运行花费时间的总和，即该模块完成一次运行所花费的时间与模块调用频率之积。因为各功能模块的负载是负荷均衡处理时的主要判断依据，所以首先需要完成对飞控系统各个功能模块的运行时间的统计。各个功能模块参数见表1。

表1 功能模块参数统计表

Table 1 Parametric statistics of function module

模块ID	功能模块	平均运行时间/μs	模块调度频率/Hz	模块负载/ms
1	GPS 模块	715	20	14.3
2	IMU 模块	1228	100	122.8
3	空速计/高度计模块	1156	50	57.8
4	数据融合模块	1983	100	198.3
5	控制律解算模块	2396	100	239.6
6	导航制导模块	1470	50	73.5
7	控制输出模块	322	100	32.2
8	故障监测与处置模块	1119	20	22.38
9	遥控接收模块	595	25	14.875
10	遥测发送模块	342	25	8.55

3.2 静态负载均衡

静态负载均衡是指在飞控系统运行的初始时刻，主节点根据各模块负载的统计结果，通过静态负载均衡算法进行计算，再将所有功能模块均衡地分配至3个解算从节点上，使得飞控系统在开始运行时能够合理地利用各节点的资源，为后续系统实现动态负载均衡奠定基础。

为了能够实现静态负载均衡，分配功能模块时，需要遵循以下两个原则。

1）功能模块完全分配原则，即各个节点功能模块的个数总和应该与全局的功能模块个数相等，不允许出现功能模块未被分配的情况。

2）节点负载均衡原则，即根据飞控软件中的功能模块负载的大小，将各个节点的负荷分布在一个负载均值范围内。假设每个功能模块的负载大小为 $M_i(i=1,2,\cdots,n)$，每个节点的负载为 $P_i(i=1,2,3)$，理想负载为 P_{ideal}，则各个节点的实际负载与理想负载和功能模块负载之间的关系为

$$P_1 \approx P_2 \approx P_3 \approx P_{ideal} = \frac{1}{3}\sum_{i=1}^{n} M_i \text{ 。} \quad (1)$$

根据上述两个原则，设计静态负载均衡算法：假设各个节点的负载为 $nodeload[i](i=1,2,\cdots,n)$，首先

对节点 1 进行分配，对所有功能模块依次进行判断，如果节点 1 的负载加上该功能模块后使节点 1 的负载与理想负载 P_{ideal} 的偏差更小，即满足式(2)时则将该模块分配至节点 1，否则不进行分配。节点 1 分配完成后，节点 2 同理。根据功能模块完全分配原则，在为节点 3 分配功能模块时，需要将所有未分配的功能模块分配至节点 3 中，这样可以避免出现部分功能模块未被分配解算节点的情况。

$$|P_{ideal} - nodeload[i] - M_j| < |P_{ideal} - nodeload[i]| \quad 。(2)$$

静态负载均衡的具体流程如图 3 所示，其中，i 为节点号，j 为功能模块 ID 号，n 为模块总个数。

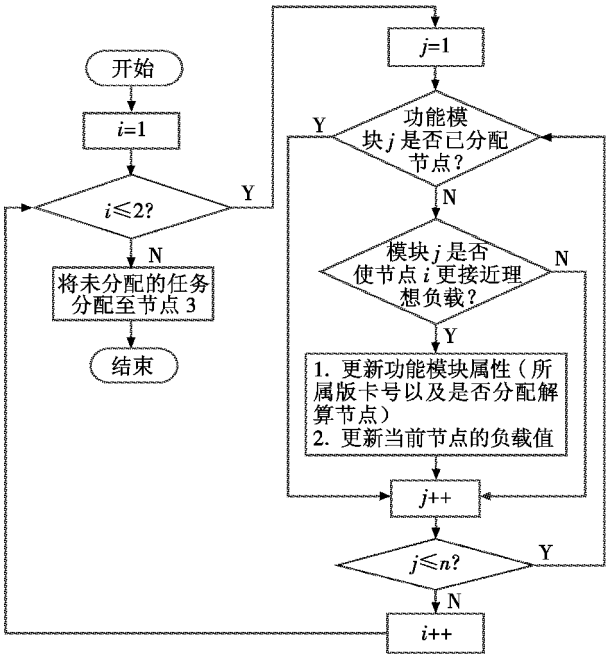

图 3　静态负载分配算法

Fig. 3　Static load allocation algorithm

3.3　动态负载均衡

随着系统运行时间的推移，无人机飞行阶段会进行切换，飞行模态会发生改变，各个功能模块的数据解算量会发生变化，同时飞行控制软件中存在事件触发型的任务，如当部分传感器受到外界干扰而发生故障时，相应的故障处置任务会被激活，从而各个节点的负荷会产生变化，与最初利用静态负载均衡划分的结果产生一定的偏差，此时就需要使用动态负荷均衡策略才能实现各个节点在运行过程中始终保持较为合理的负荷，保证系统的正常运行[8-9]。动态负载均衡主要涉及负载均衡的决策和功能模块的动态迁移。

根据国军标要求，每个处理器的 CPU 利用率不应超过 70%。当 CPU 利用率超过 70% 时，系统的稳定性会下降，需要留出一定的裕度，因此设定 CPU 利用率 U 的上限和下限分别为 60% 和 10%。超过上限的为过载节点，低于下限的则为轻载节点，在上下限之间的为中载节点，如图 4 所示。节点负载过重可能会导致相关的飞控任务不能及时得到执行，出现安全隐患，因此该节点要主动向主节点提出申请，将部分功能模块迁移至负载相对较轻的节点；同时负载较轻的节点由于执行任务较少，会造成资源浪费，因此负载较轻的节点要主动向主节点申请成为模块迁入节点，接收相应的功能模块的迁入。

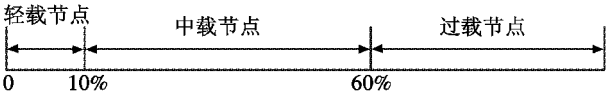

图 4　不同负载状态下的节点类型划分

Fig. 4　Node type partitioning in different load states

由于各个节点均使用 MPC565 作为主控芯片，可以认为各个节点的处理能力相当，只需要将负载最小的节点作为迁入节点即可，所以实现动态负载均衡的重点在于确定需要迁出的功能模块，这时需要考虑两方面的因素：功能模块迁移后对迁入和迁出节点负载的影响以及模块迁移对通信开销的影响。

在进行模块迁移时，可能出现迁移的模块负载过小，完成模块迁移后，当前节点仍处于过载状态，以及迁移的模块负载过大，导致迁入节点的负载大大增加，进入了过载状态的情况。为了避免上述情况的发生，在完成模块迁移后，迁出节点和迁入节点的负载比较接近是最理想的状态，即使得迁移的功能模块的负载接近于理想负载 U_0，即

$$U_0 = \frac{1}{2}(R_i^w - L_j^w) \quad i \neq j \qquad (3)$$

式中：R_i^w 表示过载节点的负载；L_j^w 表示迁入节点的负载。设迁出节点上功能模块集合为 $\{M_k \mid k \in (1,2,\cdots,n)\}$，功能模块对应的负载集合为 $\{M_k^w \mid k \in (1,2,\cdots,n)\}$，将迁出节点上的功能模块负载与理想负载的差值定义为理想负载指数 Δl_k，即

$$\Delta l_k = M_k^w - U_0 = M_k^w - \frac{1}{2}(R_i^w - L_j^w) \quad 。 \qquad (4)$$

在考虑模块负载因素的同时，还需要考虑节点功能模块与迁入节点和迁出节点之间的数据耦合关系。如图 5 所示，在对功能模块 i 进行迁移前，该模块与迁出节点的其他模块间的数据交互属于节点内的数据耦合，数值大小记为 n_i，与迁入节点中的功能模块属于节点间的数据耦合，数值大小记为 t_i，功能模块 i 迁移后，与迁出节点中功能模块之间的数据耦合大小记为 n_i'，与迁入节点中的功能模块之间的数据耦合大小记为 t_i'。

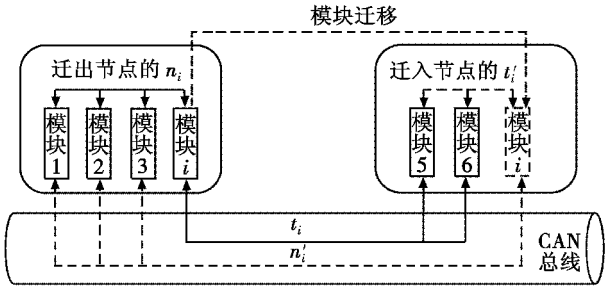

图5 数据耦合性的变化

Fig. 5 Changes in data coupling

对于节点内的数据耦合，功能模块需要的资源可以直接从软件总线的资源池中读取。但是对于节点间的数据耦合，则需要考虑节点间数据的传输和更新的时间。因此，节点内的数据耦合对系统造成的影响与节点间的数据耦合造成的影响相比可以忽略。所以仅考虑节点间的数据耦合对功能模块迁移造成的影响，则模块迁移后对系统造成的数据耦合变化为

$$\Delta c_i = n_i' - t_i \quad 。 \tag{5}$$

由于上述模块迁移过程中两个因素的量纲不一致，在进行评估时需要进行标准化处理，具体的方法为

$$l_i = \frac{|M_i^w - U_0|}{U_0} \tag{6}$$

$$c_i = \frac{n_i' - t_i}{\sum_{i=1}^{n} t_i} \quad 。 \tag{7}$$

因为上述两个因素对模块迁移产生的影响大小不同，所以利用权重因子作为在对一个功能模块进行综合评价时两个因素所占比重，分别定义模块负载所占比重和节点间通信开销所占比重为 W_{load} 和 W_{coup}，所以各个功能模块综合评价结果为

$$v_i = W_{\text{load}} * l_i + W_{\text{coup}} * c_i \tag{8}$$

式中，v_i 是模块 M_i 的综合评价结果，因此将迁出节点中功能模块的 v_i 的最小值作为迁出模块。这样的评价机制能够避免将数据耦合较大的模块分配至不同的节点，同时考虑了迁移前后负载的影响，使得飞控系统的各节点能够处于负载均衡状态。

3.4 功能模块迁移

无人机飞行控制系统具有高实时性的要求，模块迁移延迟时间是衡量迁移算法的重要指标，即延迟时间越小对系统影响越小。而传统的基于网络通信的动态迁移算法，如内存预拷贝算法，由于需要将源主机的全部状态通过反复迭代的方法传输至目标主机而产生数据重复拷贝导致总时间增加[10]。针对上述问题，本文利用 VxWorks 操作系统中提供的动态加载机制和软件总线设计了模块预加载算法。

飞行控制系统中每个节点均利用 VxWorks 操作系统的 TrueFFS 功能组件建立文件系统[11]，并在文件系统中存储所有功能模块的文件，当相应的功能模块需要运行时，利用软件总线中的模块管理器将相应的目标文件从文件系统加载至内存中执行。同时，各个功能模块运行所需的信息在各节点上均是互相备份的，因此与传统的动态迁移方法相比，当模块迁出时，不需要将相应的信息传输至目标节点，只需要利用软件总线的模块管理器卸载相应的功能模块，减少了功能模块迁移时系统资源的开销和传输的延迟。

模块预加载算法的具体流程如下：首先从迁入节点的文件系统中读取需要进行迁移的功能模块，实现预加载并进入运行状态，使能各个节点中软件总线的仲裁功能，即各个节点对内部资源池的更新仍使用迁出节点解算的结果，并向主节点发送加载完成应答信号；主节点接收到应答信号后，发送模块卸载信号至相应的迁出节点，迁出节点调用 VxWorks 操作系统提供的 unld( )库函数完成该模块的卸载后向主节点发送卸载完成信号；主节点在接收到卸载完成信号后，再在物理总线上广播仲裁功能关闭信号帧，使各个节点关闭仲裁功能，此时对资源池的更新使用迁入节点解算的结果。具体的迁移过程如图6所示。

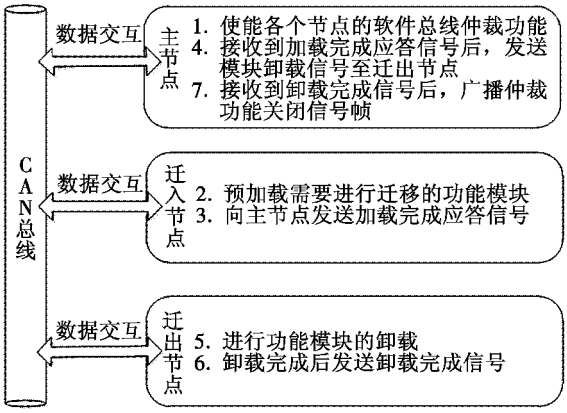

图6 功能模块迁移流程

Fig. 6 The process of functional module migration

3.5 可靠性分析

可靠性作为飞行控制系统的一项重要指标，决定了无人机飞行的安全。为了定量分析系统的可靠性，利用可靠性分析模型对样例飞行控制系统和基本三余度飞行控制系统的可靠性进行对比分析。

系统可靠性的度量称为可靠度，记作 $R(t)$。由文献[12]可知，在一个无冗余系统（即单节点系统）中，系统的可靠度可以表示为

$$R(t) = \mathrm{e}^{-\lambda t} \tag{9}$$

式中，λ 为失效率，单位为失效数/h，在系统正常生命周期中为常数。

通常采用将飞行控制系统设计为双余度和三余度

等结构的方法来提高飞行控制系统的可靠性。基本三余度结构如图 7 所示，该结构中包括 3 个解算单元和 1 个表决器，各解算单元运行同样的解算任务，表决器根据各解算单元的结果进行处理，输出最终的结果，即该结构为并－串联结构。

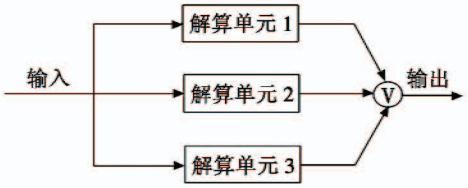

图 7　基本三余度飞行控制系统

Fig. 7　Basic triplex-redundancy flight control system

由于 3 个解算单元和表决器具有相同的结构，因此各解算单元和表决器具有相同的可靠度。假设失效率为 λ_{node}，则各节点的可靠度为 $R(t)=e^{-\lambda_{node}t}$。根据并－串联可靠度公式[13]可得该系统的可靠度为

$$R_a(t)=(1-(1-e^{-\lambda_{node}t})^3)\cdot e^{-\lambda_{node}t}。\quad (10)$$

样例飞行控制系统的可靠性分析结构如图 8 所示，该结构为并联系统结构。

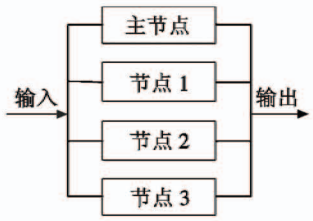

图 8　样例飞行控制系统可靠性分析结构

Fig. 8　The construction of sample flight control system for reliability analysis

假设样例飞行控制系统各节点与三余度飞控系统的解算单元结构相同，可以得到各节点的可靠度为 $R(t)=e^{-\lambda_{node}t}$。由于功能模块可以由系统中任一节点加载运行，因此根据并－串联可靠度公式可得该系统的可靠度为

$$R_b(t)=1-(1-e^{-\lambda_{node}t})^4。\quad (11)$$

根据文献[13]和工程经验，取 λ_{node} 为 10^{-5}/h，绘制了两种结构的飞行控制系统的可靠度对比曲线，如图 9 所示。

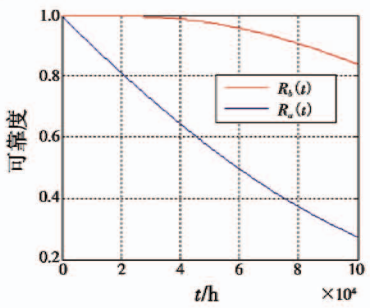

图 9　可靠度对比曲线

Fig. 9　The curves of reliability

从图 9 中可以看出，负载均衡的飞行控制系统与基本三余度结构相比，可靠度得到大幅度提高，满足飞行控制系统可靠性的要求。

4　测试验证

为了验证负荷分担策略的正确性和合理性，需要对功能模块分配结果以及各个节点的 CPU 利用率是否能够动态地处于中载状态进行测试与分析。

4.1　静态负载均衡功能测试验证

整个飞行控制系统上电后，首先进行各个节点的初始化，待所有节点初始化完成后，主节点根据表 1 中的数据执行静态负载均衡分配算法，并将功能模块分配的结果输出至各从节点，分配结果如图 10 所示。

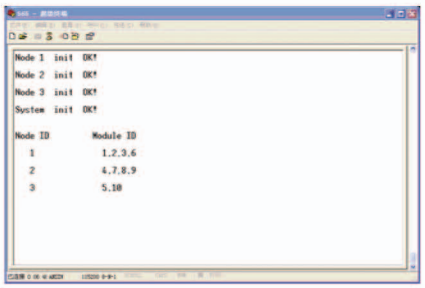

图 10　静态负载均衡分配结果

Fig. 10　The result of static load allocation

从图 10 中可以看出，节点 1 分配的功能模块分别为 GPS 模块、IMU 模块、空速计/高度计模块和导航制导模块，节点 2 分配的功能模块分别为数据融合模块、控制输出模块、故障监测与处置模块和遥控接收模块，节点 3 分配的功能模块分别为控制律解算模块和遥测发送模块。根据计算，3 个解算节点的 CPU 利用率应分别为 27.2%，26.7% 和 24.8%。

在完成静态负载均衡后，对各个节点的负载情况进行实时监测，结果如图 11 所示。由监测结果可知，各个解算节点在完成静态负载均衡后 CPU 利用率均在 26% 左右，与理论计算值相符。结果表明，负荷分担式飞行控制软件在进行静态负载均衡时能够做到准确均衡地分配各个功能模块，使系统的初始状态处于一个较为合理安全的状态。

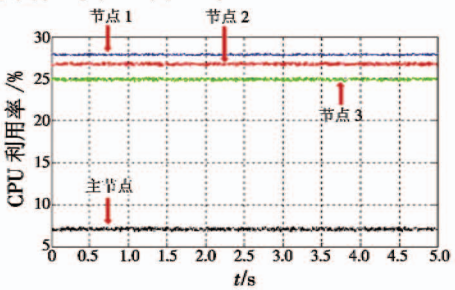

图 11　静态负载均衡分配后各节点负载情况

Fig. 11　Load of each node after static load allocation

4.2 动态负载均衡功能测试验证

为了便于验证动态负载均衡及估计各个节点的负载，同时模拟系统受到外界干扰时的情况，在静态负载均衡的基础上进行手动分配，将 IMU 模块、数据融合模块、控制输出模块以及故障监测与处置模块分配至解算节点 3。通过图 12 可以看出，完成功能模块手动分配后，节点 1 的 CPU 利用率为 14.56%，节点 2 的 CPU 利用率为 1.4%，节点 3 的 CPU 利用率为 62.3%。负荷较轻的节点 2 首先会根据主节点的调度结果，加载需要迁移的功能模块，待系统稳定运行后，过载节点 3 将迁出的功能模块卸载，此时，进行了负荷分担的 2 个节点的 CPU 利用率比较接近，均在 32% 左右，达到了预期的效果。

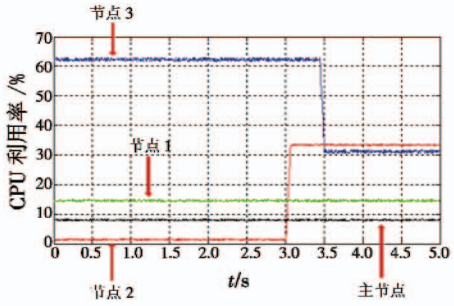

图 12 某一个节点过载时的动态负载均衡

Fig. 12 The process of dynamic load balancing when a node is overloaded

综合上述测试分析，当系统受到外界干扰时，动态负载均衡策略能够根据系统与理想负载的偏差进行调整，使得系统各节点的负载始终处于合理的状态，能够保证系统持续高效运行以及节点间的负载均衡，提高了分布式架构无人机飞行控制软件的可靠性和安全性，能满足系统的要求。

5 结语

本文结合分布式飞控系统的结构特点，设计了飞行控制软件的负荷均衡策略，最终实现了分布式架构下节点间的静态和动态负载均衡，使得整个系统能够持续高效地运行。通过测试和分析，表明系统中各个节点均能够动态地处于中载状态，多个节点能够可靠、稳定地协同工作，为分布式飞行控制软件设计提供了新思路。

参 考 文 献

[1] 孟冲. 小型无人机负荷分担式容错飞行控制软件设计[D]. 南京：南京航空航天大学，2014.

[2] 张增安，陈欣，吕迅竑. 一种用于无人机的分布式飞行控制系统设计[J]. 计算机系统应用，2010，19(8)：16-19，61.

[3] 吴建军，郑国辉，张小林. 中小型无人飞行器二余度 CAN 总线网络设计[J]. 计算机测量与控制，2012，20(3)：813-815.

[4] 张杨，于银涛. VxWorks 内核、设备驱动与 BSP 开发详解[M]. 2 版. 北京：人民邮电出版社，2011：36-42.

[5] 靳朋飞. 基于 VxWorks 的软件总线及其自主恢复技术[D]. 西安：陕西师范大学，2013.

[6] 贾振宇. 基于软总线的组件式飞行控制软件设计[D]. 南京：南京航空航天大学，2016.

[7] BRUCE-BOYE C, KAZAKOV D A. Distributed data acquisition and control by software bus[J]. 自动化博览，2004，21(5)：98-99.

[8] 刘洋，李林峰. 调度机制对 CPU-MEM 负载共享系统的性能影响[J]. 计算机工程与设计，2008，29(3)：633-638.

[9] 宁涛. 面向嵌入式应用的动态加载机制研究[D]. 重庆：重庆大学，2008.

[10] 李建彬. 基于虚拟机的分布式容灾备份技术研究[D]. 长沙：国防科学技术大学，2010.

[11] 谭明. TrueFFS 文件系统的底层结构与性能研究[D]. 长沙：国防科学技术大学，2010.

[12] 袁由光. 容错计算原理[M]. 哈尔滨：哈尔滨工程大学出版社，2005.

[13] 石贤良. 飞行控制计算机系统余度管理技术研究[D]. 西安：西北工业大学，2006.