УДК 004.9
В. В. Дук
С. С. Костєлов
Луцький національний технічний університет

ПОБУДОВА ВІДМОВОСТІЙКИХ РОЗПОДІЛЕНИХ СИСТЕМ АВТОМАТИЗАЦІЇ НА БАЗІ ОПЕРАЦІЙНОЇ СИСТЕМИ РЕАЛЬНОГО ЧАСУ QNX

В статті описано спосіб організації надійних розподілених систем автоматизації з використанням операційної системи реального часу QNX. Проведено аналіз можливостей вбудованої підсистеми Qnet.

Ключові слова: *розподілені обчислення, кластерні системи, операційна система реального часу QNX.*

Стрімкий розвиток інформаційних технологій за останні 10 років суттєво вплинув на коло завдань, які постають перед обчислювальними системами та системами автоматизації зокрема. Виконання завдань із складними алгоритмами роботи на сьогодні потребує використання не тільки багатопроцесорних, але і багатомашинних (кластерних) систем. Відповідно такі системи збільшують продуктивність роботи, але потребують особливої уваги при проектуванні, для забезпечення надійсності, відмовостійкості (*failover*) та балансування навантаження (*load balancing*).

Використання розподілених систем на сьогодні є досить актуальним. Незважаючи на це, наукових праць, які б детально розглядали використання кластерних систем при проектуванні систем автоматизації є досить небагато. Тому виникла потреба у дослідженні способів реалізації багатомашинних обчислень з використанням операційної системи реального часу QNX. Метою дослідження є відображення практичного способу організації кластерної обробки даних, яка використовує специфічні властивості операційної системи.

Під поняттям «кластер» розуміють групу обчислювальних машин, які зв'язані між собою та функціонують як єдиний вузол обробки інформації.

До загальних вимог, які постають перед кластерними системами, можна віднести наступне:

1. Висока швидкодія.
2. Спільний доступ до ресурсів.
3. Висока доступність.
4. Масштабованість.
5. Зручність та швидкість в обслуговуванні та управлінні.

Однак не кожен обчислювальний процес дає певні переваги при використанні паралельних обчислень. Для того, щоб отримати результати в прирості продуктивності алгоритм процесу повинен мати певні властивості паралелізму, циклічного виконання а також мінімальну залежність наступних циклів від результатів попередніх циклів (ітераційна властивість). Можна визначити такі основні завдання автоматизації, які легко адаптовуються для паралельних обчислень:

1. Обробка, аналіз та ідентифікація зображень.
2. Пошук інформації у великих об'ємах даних, використовуючи складні ключові умови.
3. Створення надійних систем баз даних.
4. Множинні обчислення цільової функції.
5. Організація моніторингу та контролю параметрів великої кількості давачів.
6. Керування складними розподіленими виробничими лініями.
7. Генерація незалежних керуючих сигналів та повідомлень.

Операційна система реального часу QNX Neutrino (QNX 6) за своєю суттю призначена для використання у складних, зазвичай розподілених та навантажених, системах автоматизації, які мають високі вимоги до надійності та доступності, а системний збій яких може призвести до

масштабних матеріальних збитків або техногенних катастроф. Для забезпечення високої надійності автоматизованих систем, в QNX на рівні з мікроядерною архітектурою та повною ізоляцією модулів в ОЗП, додатково передбачено ряд механізмів, серед яких можна виділити наступні [1]:

1. Механізм адаптивного квотування ресурсів ЦП та ОЗП.
2. Технологія швидкої активізації пристроїв.
3. Механізм балансування навантаження на сервіси.
4. Механізм формування роз приділеного обчислювального середовища.
5. Засоби підтримки резервування фізичних каналів зв'язку в кластері

Оскільки операційна система QNX є повністю сумісною з стандартом POSIX, вона підтримує стандартні методи міжпроцесної взаємодії (*IPC*), які можуть бути використані не тільки на локальному вузлі, але і при взаємодії з зовнішніми вузлами (табл. 1).

Таблиця 1.

POSIX-сумісні методи міжпроцесної взаємодії в операційній системі QNX.

Назва сервісу	**Рівень реалізації**
Синхронні повідомлення (Message passing)	Мікроядро
Імпульси (Pulses)	Мікроядро
Сигнали (Signals)	Мікроядро
Черги повідомлень (Message queues)	Менеджер черг (mqueue)
Розділювана пам'ять (Shared memory)	Менеджер процесів
Неіменовані канали (pipes)	Менеджер каналів (pipe)
Іменовані канали FIFO (File Input File Output)	Менеджер файлової системи

Відповідно, використовуючи стандартні методи міжпроцесної взаємодії та базуючись на мережевому стеку протоколів TCP/IP, організація кластерних обчислень в QNX може бути аналогічною до їх реалізації в операційних системах Linux/Unix або Windows. Проте головною перевагою операційної системи QNX є наявність спеціальної підсистеми для кластерної взаємодії. Механізм розподілених обчислень TDP (*Transparent Distributed Processing*) операційної системи QNX базується на протоколі Qnet, що працює на транспортному (четвертому) рівні мережевої моделі OSI. Цей протокол дозволяє ефективно обмінюватися ресурсами та сервісами між вузлами мережі в «прозорому» режимі, з мінімальними накладними втратами продуктивності.

Для забезпечення максимальної доступності та рівномірного розподілу навантаження технологія QNX TDP підтримує багатократні з'єднання між процесорами. Якщо одне із встановлених з'єднань стає недоступним, то дані будуть автоматично перенаправлені через інші доступні з'єднання. TDP також надає можливість розподілу навантаження між усіма доступними з'єднаннями, що суттєво збільшує продуктивність всієї системи.

Можливості кластерних обчислень в мережі Qnet дозволяють виконувати наступні завдання [2]:

1. Віддалено використовувати файлову систему інших вузлів.
2. Використовувати службу віддаленого виклику процедур мережі Qnet.
3. Створювати програмні рішення, де одночасно використовується декілька процесорів (ЦП), які взаємодіють за допомогою механізму обміну повідомленнями.
4. Розділяти виконання завдання між декількома ЦП, де кожен процес виконує окрему функцію. Процеси координують свою роботу за допомогою механізму обміну повідомленнями.
5. Переносити виконання програми з одного ЦП чи симетричної мультипроцесорної системи на декілька однопроцесорних машин та розділяти виконання процесів між ЦП.

Для виявлення та розпізнавання вузлів в мережі Qnet використовується механізм NDP (*Node Discovery Protocol*), який дозволяє автоматично реєструвати нові вузли в мережі, використовуючи широкомовні повідомлення (*broadcast*). Це дуже зручний і динамічний спосіб, який не вимагає втручання користувача для того, щоб додати новий вузол мережі. Можливі варіанти з використанням виявлення на базі DNS (стек протоколів TCP/IP), або пошук вузлів у статичному файлі (*file*).

Також потрібно зауважити, що, використовуючи протокол Qnet, маніпуляція файлами на віддалених машинах здійснюється звичним програмним забезпеченням (*ls, cat, touch, mv, cp, rm*) у такий самий спосіб, якби вони знаходилися на локальному вузлі (*рис. 1*).

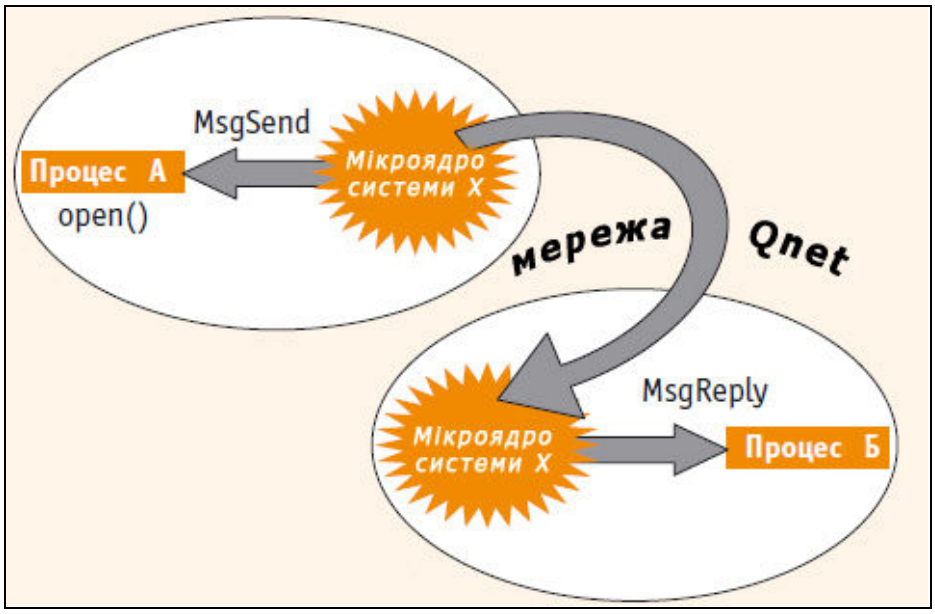

Рис. 1. Схема міжсистемної взаємодії в мережі Qnet.

Програмний код, який потрібен для клієнт-серверної взаємодії ідентичний до того, який використовується у випадку з одним вузлом, із вказаням важливого параметра - повної назви шляху (*pathname*). Назва шляху складається з префіксу, який вказує вузол (*node*) на якому працює та чи інша служба (наприклад: */dev/socket*). Цей префікс використовується для отримання дескриптора вузла (*node descriptor*), за допомогою якого викликається низькорівнева процедура ядра *ConnectAttach()*. Кожен вузол в мережі має свій унікальний дескриптор. Приклад відображення віддаленого вузла *lab1*:

/net/lab1/dev/socket
/net/lab1/dev/ser1
/net/lab1/home
/net/lab1/bin

В якості прикладу розглянемо роботу функції *open()*. На тестовому стенді основний вузол (*server*) буде використовувати послідовний порт (*/dev/ser1*) дочірнього вузла (*lab1*). На вузлі *server* виконується функція *open()*, вказуючи шлях */net/lab1/dev/ser1* .

```
fd = open("/net/lab1/dev/ser1", O_RDWR);
read (fd, &buffer, sizeof(buffer));
```

Алгоритм виконання операцій (*рис. 2*):

1. Виконується передача повідомлення від клієнта локальному менеджеру процесів (*npm-qnet*), для розпізнання мережевого шляху */net/lab1/dev/ser1*. Локальний менеджер процесів поверне

повідомлення з перенаправленням, вказуючи що клієнт повинен звернутися до менеджера мережі, так як вказано простір імен "*/net*" .

2. Клієнт передає повідомлення локальному менеджеру мережі, знову запитуючи інформацію про наступний крок з'єднання. Локальний менеджер мережі повертає повідомлення з перенаправленням, вказуючи дескриптор вузла, ідентифікатор процесу та ідентифікатор каналу менеджера процесів на вузлі *lab1*.
3. Клієнт відкриває з'єднання з менеджером процесів на вузлі *lab1*, знову запитуючи інформацію про наступний крок з'єднання. Менеджер процесів на вузлі *lab1* повертає повідомлення з перенаправленням, де вказано дескриптор вузла, ідентифікатори каналу та процесу драйвера послідовного інтерфейсу, що знаходяться уже на його власному вузлі .
4. Клієнт відкриває з'єднання з драйвером послідовного інтерфейсу на вузлі *lab1*, та отримує ідентифікатор, який можна використовувати для надсилання та отримання повідомлень. Тепер усі повідомлення між клієнтом та сервером є прямими і здійснюються аналогічно до їх локального варіанту.

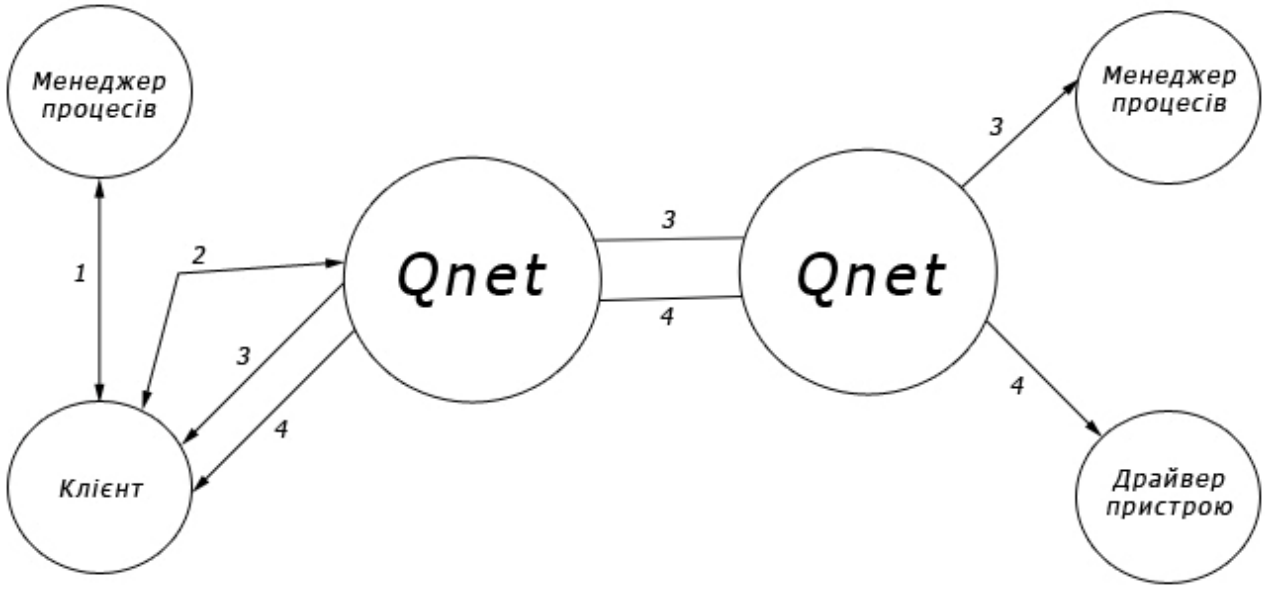

Рис. 2. Алгоритм роботи функції *open()* в мережі Qnet.

При проектуванні систем реального часу часто виникає питання про якість обслуговування або якість сервісу (*QoS - Quality of Service*). У контексті Qnet, QoS зводиться до вибору каналу передачі даних. У системі з двома або більше мережевими адаптерами, Qnet буде проводити вибір згідно з вказаною QoS-політикою. Надійність функціонування мережі Qnet забезпечується за допомогою резервування фізичних ліній зв'язку. Для кожного логічного з'єднання процесу-клієнта з процесом-сервером можна визначити один з трьох варіантів вибору фізичного каналу:

1. З автоматичним балансуванням наватаження (*loadbalance*). В даному випадку Qnet самостійно приймає рішення, яким з доступних фізичних ліній зв'язку передавати пакети даних. Для вибору мережевого адаптера Qnet аналізує час відклику віддаленого вузла на запити, які надсилаються через доступні фізичні канали. Для зв'язку використовується та лінія, яка в даний момент часу є найшвидшою. Цей спосіб передачі даних використовується системою за замовчуванням.
2. З пріоритетом (*preferred*). В даному варіанті вказується, який з мережевих інтерфейсів є пріорітетнішим для передачі даних. У випадку, якщо пріоритетний мережевий адаптер недоступний, Qnet буде використовувати решту мережевих адаптерів, використовуючи

автоматичне балансування навантаження між ними, тобто система переходить у режим балансування навантаження (loadbalance).

3. Ексклюзивний (*exclusive*). Дозволяє задавати передачу даних строго через визначений адаптер. Якщо вказаний адаптер стане недоступним, Qnet взагалі не буде передавати дані, навіть якщо віддалений вузол доступний через інші мережеві адаптери.

Варіанти вибору фізичної лінії для різних логічних з'єднань можна задавати на будь-яких етапах дії циклу автоматизованої системи: під час розробки, встановлення, або під час експлуатації. Внесення змін не потребує повторної компіляції програми, що також є важливим фактором для сертифікованого програмного забезпечення. Для того щоб визначити певну QoS-політику для логічного з'єднання, необхідно модифікувати шлях наступним чином:

/net/lab1~preffered:en0/dev/ser1

Параметр QoS завжди починається з символу ~ (тильда), після якого йде назва політики вибору фізичного каналу, та назва мережевого адаптера через двокрапку. Протокол Qnet дозволяє проводити інкапсуляцію повідомлень для передачі в мережі IP (стек TCP/IP), що дає змогу маршрутизувати Qnet-пакети та передавати їх в віддалені мережі. Це дає змогу для комунікації з географічно віддаленими вузлами.

Таким чином, з впевненістю можна сказати, що протокол Qnet операційної системи QNX є достатньо потужним засобом, який можна використати при проектуванні автоматизованих систем. Використовуючи Qnet, можна з легкістю масштабувати існуючі системи автоматизації, швидко нарощувати обчислювальні потужності та розширювати спектр виконуваних завдань.

Література:

1. Операционная система реального времени QNX Neutrino 6.3. Руководство пользователя: Пер. с англ. - СПб.: БХВ-Петербург, 2009 - 480 с.: ил.
2. Операционная система реального времени QNX Neutrino 6.3. Системная архитектура: Пер. с англ. - СПб.: БХВ-Петербург, 2006 - 336 с.: ил.