УДК 004.416.3
DOI: 10.15827/0236-235X.128.655-664
Дата подачи статьи: 29.05.19
2019. Т. 32. № 4. С. 655–664

Особенности портирования Robot Operating System на программно-аппаратную платформу семейства «Эльбрус»

А.А. Тачков [1], *к.т.н., начальник отдела «Автоматизированные транспортные системы», tachkov@bmstu.ru*
А.Ю. Вуколов [1], *программист, twdragon@bmstu.ru*
А.В. Козов [1], *инженер, alexey.kozov@gmail.com*

[1] *НУЦ «Робототехника» МГТУ им. Н.Э. Баумана, 105037, г. Москва, Россия*

Наиболее распространенным вспомогательным фреймворком, облегчающим разработку систем управления мобильными роботами, является ROS (Robot Operating System), однако его полноценная поддержка только для операционных систем Ubuntu/Debian Linux приводит к ограничению возможности использования вычислительных средств отечественного производства в составе проектируемых систем управления. Авторами статьи было осуществлено портирование ROS версии Melodic Morenia на отечественную программно-аппаратную платформу «Эльбрус» (вычислительный комплекс на базе процессора «Эльбрус-4С»).

В данной работе рассмотрены основные особенности процесса портирования, связанные с отличиями операционной системы «Эльбрус» от большинства существующих дистрибутивов Linux, а также согласование различающихся между собой версий имеющегося на платформе «Эльбрус» и требуемого в ROS программного обеспечения. Так как часть используемых при сборке ROS библиотек имеют зависимости, полностью удовлетворяемые на целевой платформе, данные библиотеки были упакованы в deb-пакеты для повторного применения на аналогичных вычислительных комплексах. Кроме того, разработаны сценарии автоматизированной сборки и развертывания подготовленного к работе ROS.

В статье описано тестирование работоспособности ROS на программно-аппаратной платформе «Эльбрус» применительно к задаче построения многослойной карты проходимости системой управления мобильного робототехнического комплекса на основе представленных облаком точек данных от сканирующих дальномеров. Приведены сравнительные результаты по временным интервалам обработки облака точек классификаторами, а также по времени обновления слоев карты, полученные для одной и той же версии ROS на вычислительных комплексах на базе процессоров «Эльбрус-4С», Intel Core i3-3220 и Intel Core i7-6700HQ. Сделан вывод о полной работоспособности ROS Melodic Morenia при развертывании на программно-аппаратной платформе «Эльбрус».

***Ключевые слова**: программно-аппаратная платформа «Эльбрус», ROS, портирование, сборка программных пакетов, тестирование производительности, многослойная карта, облако точек, мобильный робот.*

Одним из факторов, сдерживающих развитие систем управления отечественными автономными мобильными роботами, является отставание отечественной элементной базы от зарубежной [1, 2]. В связи с этим оснащение систем управления вычислительной техникой, созданной на базе отечественных микропроцессоров, и ПО отечественной разработки остается нерешенной проблемой. В части аппаратной составляющей систем управления произошел существенный сдвиг: компанией МЦСТ (г. Москва) были разработаны перспективные отечественные аппаратные платформы на базе многоядерных процессоров семейства «Эльбрус» («Эльбрус-4С», «Эльбрус-8С») [3], которые могут быть использованы в задачах мобильной робототехники [4, 5]. Данная аппаратная платформа работает под управлением *операционной системы* (ОС) «Эльбрус», имеющей пока весьма ограниченное ПО, предназначенное для разработки систем управления роботами.

Мировой тенденцией в области разработки ПО для автономных мобильных роботов является применение Robot Operating System (ROS). ROS – это свободно распространяемый фреймворк для программирования роботов. Он не является ОС в привычном смысле, но обеспечивает определенную функциональность, характерную для ОС: абстрагирование от аппаратных средств, низкоуровневое управление устройствами, реализацию библиотечных

функций, передачу сообщений между процессами и управление пакетами. ROS спроектирован как слабо связанная система, в которой одновременно выполняется множество процессов. Каждый процесс, называемый узлом, должен отвечать за одну задачу. Узлы общаются друг с другом, используя форматированные на бинарном уровне сообщения, проходящие через именованные каналы, называемые темами. Каждый узел может отправлять или получать данные от другого узла, используя темы, а также синхронные (сервисы) и асинхронные (действия) интерфейсы и модели событий, предоставляемые мастер-процессом. Передача данных полностью абстрагирована как от машины, на которой запущен узел, так и от интерфейсов связи, включая сеть. Для обеспечения единообразного функционирования узлов в ROS введен собственный подход к объектно-ориентированной разработке, основанный на специальном шаблоне проектирования «издатель–подписчик» [6]. При этом во время сборки проекта специфичным для ROS процессором сценариев *catkin* производится генерация шаблонного исходного текста на нескольких целевых языках (включая Python, CommonLISP, Java, C/C++).

Преимуществом ROS является поддержка большого числа драйверов, позволяющих единым образом работать с периферийными устройствами роботов (GPS/GLONASS-приемниками, сканирующими лазерными дальномерами, видеокамерами, тепловизорами). Кроме того, имеются так называемые метапакеты, реализующие различные прикладные задачи робототехники, такие как навигация, построение карт, планирование траекторий движения и т.п.

Следует отметить, что на момент написания статьи полноценная поддержка последней версии ROS Melodic Morenia обеспечивается только для ОС Ubuntu и Debian на платформах AMD64, ARMHF, ARM64. Для адаптации ПО систем управления роботами к отечественным вычислительным средствам авторами статьи в инициативном порядке было осуществлено портирование ROS на программно-аппаратную платформу архитектуры VLIW (very long instruction word – очень длинная машинная команда) семейства «Эльбрус».

Особенности ПО платформы «Эльбрус»

В архитектуре VLIW задача распределения независимых команд по исполнительным устройствам возложена на компилятор. VLIW-команда содержит сразу несколько операций для параллельного выполнения суперскалярным ядром процессора. Например, ядро процессора «Эльбрус-8С» имеет шесть 64-битных FMA (fused multiply-add – совмещенное умножение-сложение) устройств, что позволяет выполнять до 12 операций над 64-разрядными числами с плавающей точкой за такт [7]. Это ключевое отличие процессоров семейства «Эльбрус» от процессоров Intel и AMD, в которых загрузку множественных исполнительных устройств обеспечивает ядро процессора за счет возможности внеочередного исполнения команд.

Задача портирования стороннего ПО на платформу «Эльбрус» облегчается тем, что в качестве базовой спецификации, описывающей правила сборки, подготовки и развертывания, используется спецификация дистрибутива Debian Linux. В рамках этой спецификации обеспечивается высокая степень интеграции ПО в систему. В частности, правила и концепции, реализованные окружением ОС, правила размещения файлов X Desktop Group (XDG) Directory Specification [8] и совместимость с большей частью интерфейсов стандарта POSIX (Portable Operating System Interface – переносимый интерфейс ОС) дают возможность настраивать условия хранения и вызова программ отдельно от устанавливаемых в систему менеджером пакетов apt [9]. Именно за счет применения XDG Directory Specification авторам работы удалось достичь изоляции дерева каталогов сборки и «замораживания» версий специализированных сторонних библиотек, используемых ROS версии Melodic Morenia. Прочие библиотеки изначально планировалось подгрузить из официального репозитория пакетов ОС «Эльбрус». Однако использовать официальный репозиторий в качестве источника стороннего ПО для ROS в полной мере не удалось. Это связано с тем, что система именования deb-пакетов, используемая компанией МЦСТ, не совпадает с таковой в большинстве основанных на спецификациях Debian дистрибутивов Linux, поэтому для портирования ROS потребовалось вносить ряд исправлений в управляющие сценарии. Некоторые библиотеки, например libnetpbm, при сборке в deb-пакет для платформы «Эльбрус» полностью утратили совместимость на уровне имен со своими аналогами с x86-совместимых платформ. При выявлении таких случаев принималось решение о подготовке собственного отдельного пакета, соответствующего требованиям лицензии

на исходный текст библиотеки, с потенциалом его дальнейшего распространения, как свободного, так и в составе ОС «Эльбрус».

Кросс-компиляция при портировании ROS на платформу «Эльбрус» не применялась, так как трудозатраты на развертывание необходимого окружения базовой машины с обеспечением доступа к нему процессора сценариев *catkin* оказались несопоставимыми с трудозатратами на сборку компонентов ROS непосредственно на *вычислительном комплексе* (ВК) «Эльбрус».

Анализ зависимостей и подготовка среды сборки

Анализ зависимостей при подготовке к портированию сводился к выделению имен библиотек и заголовочных файлов, используемых при сборке компонентов ROS. Затем проводился анализ содержимого системных каталогов ВК «Эльбрус». Таким образом, была выявлена группа зависимостей, удовлетворяемых заранее портированными пакетами из репозитория ОС «Эльбрус» (табл. 1).

Таблица 1

Сборочные зависимости ROS, имеющиеся в пакетах официального репозитория ОС «Эльбрус»

Table 1

ROS build dependency libraries deployed in official Elbrus OS package repository

Библиотека	Системное имя (Debian/Ubuntu)	Системное имя (ОС «Эльбрус»)
Eigen3	/usr/local/lib/pkgconfig/eigen3.pc	/usr/lib64/pkgconfig/eigen3.pc
libgfortran	/usr/lib/libgfortran.so	/usr/lib64/libfcc.so.2
zlib	libz-dev	libz
Perl	perl	perl 5
CFFI	libcffi	cffi
Java	libjava-dev	libjava6
YAML	libyamlcpp-dev	libyaml-cpp
Expat XML	libexpat-dev	libexpat
XML2	libxml2-dev	libxml2
X11 Server	libx11-*, libxkb-*	libx11
PNG	libpng-dev	libpng
JPEG	libjpeg-dev	libjpeg
TIFF	libtiff-dev	libtiff

Таблицы имен и файловых путей некоторых заранее портированных библиотек (табл. 1, столбец 3) не совпадали с таблицами, имеющимися в составе Linux-дистрибутивов Debian и Ubuntu (табл. 1, столбец 2), что приводило к необходимости, с одной стороны, внесения изменений в исходные тексты ROS, а с другой – создания в системных каталогах символических ссылок, экспортирующих требуемые имена. Поскольку созданные ссылки могли оказать влияние на вновь устанавливаемые пакеты и программное окружение ОС ВК «Эльбрус», использовалось описанное выше решение, заключающееся в изоляции сборки ROS в отдельном XDG-дереве, локализованном в подкаталоге /util корневой файловой системы. Затем созданное дерево каталогов подключалось к пользовательской среде через переменные окружения bash. Такой вариант развертывания обладает несомненным преимуществом: при необходимости позволяет производить оперативное подключение и отключение от рабочей среды ROS в процессе эксплуатации ВК.

Несовпадение версий библиотек и программ, имеющихся в системе и требуемых в ROS, значительно затрудняло портирование. В некоторых случаях требовалась ручная установка ПО необходимой версии, дублирующего имеющееся в системе. В частности, потребовалось ручное развертывание в созданном XDG-дереве макропроцессора CMake версии 3.7.0, так как версия CMake 3.4.3 из репозитория ОС «Эльбрус» не удовлетворяла требованиям ROS. Отдельного решения потребовала проблема отсутствия API стандартной библиотеки функций интерпретатора языка программирования Python версии 3. Для получения доступа к требуемым заголовочным файлам была отдельно развернута изолированная отладочная версия интерпретатора Python версии 3.5.5.

Проведенный анализ зависимостей сопровождался исследованием возможности полной или частичной автоматизации развертывания ROS на ВК «Эльбрус». Желаемым вариантом такой автоматизации является получение на выходе готовых deb-пакетов подобно рабочему процессу, описанному в работе [10]. Однако в силу особенностей целевой платформы полная автоматизация сборки таких пакетов оказалась невозможной. В результате проделанной работы были созданы взаимосвязанные сценарии сборки и установки зависимостей, подготовлены набор deb-пакетов библиотек ROS и сценарии сборки непосредственно ROS на ВК «Эльбрус». Представим фрагмент одного из сценариев установки зависимости на примере интерпретатора языка Python версии 3.5.5:

```
   PYVERSION=`python3 --version | sed -
e 's/^.*\ //'`
   wget https://www.python.org/ftp/
python/${PYVERSION}/Python-
${PYVERSION}.tgz
   tar -xvf "Python-${PYVERSION}.tgz"
```

```
   cd "Python-${PYVERSION}"
   mkdir build && cd build
   CXX=lcc ../configure --pre-
fix=/util/python3 --with-system-ffi --
with-system-expat
   make -j4
   make install
   ln -s /util/python3/include/
python3.5m /util/python3/include/
python3.5
```

Вся совокупность средств автоматизации сборки объединена в проект под рабочим названием ROS-Elbrus Deployment System. Проект включает в себя следующие компоненты:

– хранилище системных переменных, подготавливающее окружение bash к развертыванию;

– скрипт обеспечения удаленной аутентификации по URL и хранения реквизитов доступа к репозиториям git;

– средство обеспечения базовой интерактивности, осуществляющее прием реквизитов доступа от пользователя и управление перезагрузкой ВК в процессе развертывания ROS;

– скрипт вставки путей, добавляющий адреса дополнительного XDG-дерева каталогов ROS в списки путей для обработки по умолчанию в системе:

```
   if [[ ! -d "${EXT_ROOT}" ]]
   then
      mkdir "${EXT_ROOT}"
   fi
   if [[ ! -d "${MCST_REPO}" ]]
   then
      mkdir "${MCST_REPO}"
   fi
   mkdir "${EXT_LIB}" &&  mkdir "${EXT_
INCLUDE}" &&  mkdir "${EXT_PKGS}"
   cp -v /etc/profile /etc/profile.old
&&  cp -v /root/.bashrc /root/.bashrc.
old
   cat "${RUN_ROOT}/path-injection.
head" > /etc/profile
   cat "${RUN_ROOT}/path-injection.
head" > /root/.bashrc
   cat /etc/profile.old >> /etc/profile
&& cat /root/.bashrc.old >> /root/.
bashrc
   cat "${RUN_ROOT}/path-injection.
tail" >> /etc/profile
   cat "${RUN_ROOT}/path-injection.
tail" >> /root/.bashrc
   cp -v /etc/ld.so.conf /etc/ld.so.
conf.old
   sed -i "1i\
   ${EXT_LIB}\
   ${ROS_ROOT}/lib\
   " /etc/ld.so.conf && ldconfig
```

– набор архивов исходного текста, хранящий требующиеся для запуска ROS системные библиотеки в строго фиксированных версиях (в том числе Boost, TinyXML2, OpenSSL, Bullet3);

– набор deb-пакетов, включая NetPBM-sf-10.85.99, HDF5-1.8.21, QHull-2015.2, FLANN-1.9.1, PCL-1.9.1, с конфигурационным файлом, хранящим последовательность их установки.

Время сборки ROS Melodic Morenia с помощью разработанных средств автоматизации вместе со всеми зависимостями на ВК с процессором «Эльбрус-4С» составило около шести часов.

Описание процедуры тестирования ROS на программно-аппаратной платформе «Эльбрус»

Работа развернутого на программно-аппаратной платформе «Эльбрус» ROS была протестирована с помощью ПО системы управления мобильного *робототехнического комплекса* (РТК). Одна из базовых задач системы управления заключается в формировании используемой для планирования автономного перемещения РТК многослойной карты проходимости по данным *информационно-измерительной системы* (ИИС). В качестве основного средства ИИС используется трехмерный сканирующий лазерный дальномер (3D-лидар Velodyne HDL-32E), в качестве вспомогательного средства выступает двухмерный сканирующий лазерный дальномер (2D-лидар Hokuyo UXM-30LX-EW). Каждый слой карты является результатом работы отдельного узла ROS – классификатора, выполняющего проецирование полученного от лидара облака точек на соответствующие ячейки карты, обработку массива спроецированных точек и соотнесение каждой ячейки с одним из предопределенных классов, например, «препятствие», «большая неровность», «ровный участок» и т.п.

При тестировании портированного ROS использовались три классификатора. Два из них обрабатывают облако точек 3D-лидара: первый (классификатор препятствий, К1) выделяет опасные препятствия [11, 12], второй (классификатор профильной проходимости, К2) выполняет оценку профильной проходимости. Третий классификатор (классификатор препятствий, К3) работает с облаком точек 2D-лидара. Классификаторы передают данные в виде специализированных сообщений ROS на узел, предназначенный для обновления и перемещения многослойной карты вместе с РТК.

Реализация многослойной карты аналогична реализации, приведенной в работе [13]. Карта состоит их четырех слоев: слоев С1–С3, обновляемых по данным классификаторов К1–К3 соответственно, и сверточного слоя С4. Каждый слой карты имеет размер 800×800 ячеек, ячейка занимает в памяти один байт. Каждая ячейка <*i, j*> слоя С4 представляет собой результат байесовского вывода по данным соответствующих <*i, j*> ячеек остальных слоев с учетом доверительных вероятностей к сообщениям каждого из классификаторов по каждому из предопределенных классов. Обновление слоя С4 выполняется каждый раз последовательно для всех ячеек карты (800×800) в однопоточном режиме.

Тестирование носило комплексный характер ввиду проверки как работоспособности ROS (развертывание узлов, создание вычислительного графа, работа модели взаимодействия «подписчик–издатель», публикация тем, корректность построения дерева переходов систем координат, работа с файлами сохраненных данных – BAG-файлами ROS), так и оценки производительности процессора ВК.

Тестирование работоспособности проводилось на специальном стенде (рис. 1). Мастер-процесс ROS был запущен на ВК с процессором Intel Core i3 под управлением ОС Ubuntu 18.04. На этом же комплексе было запущено воспроизведение специального BAG-файла с записанными в полигонных условиях на РТК облаками точек от 2D- и 3D-лидаров. Ведомые узлы-классификаторы, а также узел, отвечающий за формирование многослойной карты проходимости (конфигуратор карты), запускались на ВК «Эльбрус-4С» (4 ядра, 680 МГц). Соединение ВК между собой и обмен данными между ними через темы ROS в соответствии с вычислительным графом осуществлялись по локальной сети с помощью сетевого концентратора. Процесс построения многослойной карты был визуализирован на ВК мастер-процесса ROS с помощью пакета RViz [14].

После проверки работоспособности портированного ROS было проведено тестирование

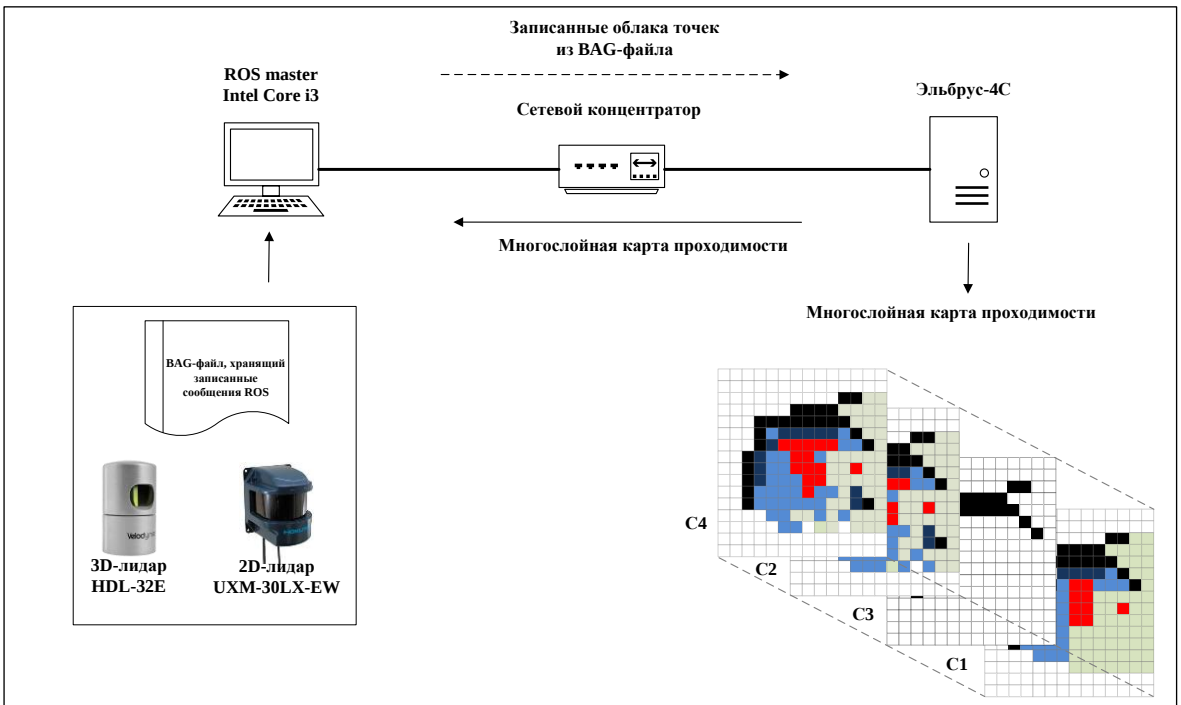

Рис. 1. Схема стенда, используемого при тестировании портированного на ВК «Эльбрус» ROS: С1 – слой классификатора препятствий по данным 3D-лидара, С2 – слой классификатора профильной проходимости по данным 3D-лидара, С3 – слой классификатора препятствий по данным 2D-лидара, С4 – сверточный слой

Fig. 1. Block diagram of the test rig for ROS portage on Elbrus platform: C1 – obstacle classification layer for 3D LIDAR input, C2 – passability classification layer for 3D LIDAR input, C3 – obstacle classification layer for 2D LIDAR input, C4 – convolutive layer

производительности ПО системы управления мобильного РТК на ВК с различными процессорами. При тестировании выполнялись замеры времени обработки одного облака точек классификаторами К1 и К2 и времени, затрачиваемого на обновление слоев С1–С4 карты. Для сравнения производительности аналогичная процедура тестирования была запущена на ВК с процессорами Intel Core i3-3220 (2 ядра, частота до 3300 МГц) и Intel Core i7-6700HQ (4 ядра, частота до 3100 МГц) и ОС Ubuntu 18.04. При тестировании на ВК с процессорами Intel запуск мастер-процесса, воспроизведение BAG-файла и визуализация осуществлялись на том же ВК, что и работа узлов-классификаторов и конфигуратора карты. Для более объективного сравнения результатов на всех ВК использовалась одна и та же версия ROS – Melodic Morenia.

Результаты сравнения производительности процессоров «Эльбрус» и Intel при тестировании ROS

В результате сравнительного тестирования ROS на ОС «Эльбрус» и Ubuntu была установлена идентичность в работоспособности ПО системы управления мобильного РТК в части построения многослойной карты проходимости, ее визуализации и обмена информацией между узлами.

Сводные результаты производительности различных процессоров («Эльбрус-4С», Intel Core i3-3220, Intel Core i7-6700) при использовании ROS представлены в таблице 2 и на рисунках 2–4.

Как видно из рисунка 2, среднее время обработки облаков точек классификаторами К1, К2 на процессоре «Эльбрус» в 8–10 раз больше, чем на процессорах Intel. Полученные результаты вполне коррелируют с соотношениями тактовых частот принимавших участие в тестировании процессоров (3100/680–3300/680). Меньшая производительность «Эльбрус-4С» объясняется тем, что ряд библиотек, используемых ROS, например Point Cloud Library, базирующаяся на библиотеке линейной алгебры Eigen, отсутствуют в репозитории ОС «Эльбрус», а в качестве высокопроизводительной математической библиотеки используется собственная библиотека eml, которая напрямую не интегрируется в ROS.

Таблица 2

Результаты сравнения производительности

Table 2

Performance comparison

ВК на базе процессора	Среднее время $t_{ср}$, с	СКО	Максимальное время t_{max}, с	Минимальное время t_{min}, с
Классификатор препятствий (К1)				
Эльбрус-4С	0,276	0,043	0,520	0,195
Intel Core i3	0,032	0,006	0,189	0,020
Intel Core i7	0,035	0,011	0,063	0,020
Классификатор профильной проходимости (К2)				
Эльбрус-4С	0,147	0,021	0,268	0,116
Intel Core i3	0,016	0,003	0,025	0,010
Intel Core i7	0,015	0,007	0,038	0,008
Обновление карты (слой С1)				
Эльбрус-4С	0,065	0,019	0,364	0,036
Intel Core i3	0,012	0,005	0,086	0,009
Intel Core i7	0,009	0,011	0,044	0,001
Обновление карты (слой С2)				
Эльбрус-4С	0,025	0,010	0,291	0,011
Intel Core i3	0,003	0,005	0,038	0,001
Intel Core i7	0,003	0,007	0,049	< 0,001
Обновление карты (слой С3)				
Эльбрус-4С	0,048	0,016	0,350	0,006
Intel Core i3	0,038	0,015	0,123	0,001
Intel Core i7	0,039	0,016	0,115	0,001
Обновление карты (слой С4)				
Эльбрус-4С	1, 465	0,093	1,781	1,325
Intel Core i3	0,155	0,012	0,272	0,134
Intel Core i7	0,153	0,012	0,188	0,124

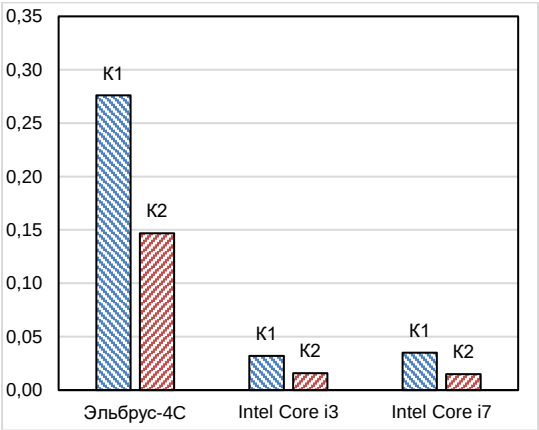

Рис. 2. Среднее время обработки одного облака точек классификаторами на различных ВК (в секундах): К1 – классификатор препятствий, К2 – классификатор профильной проходимости

Fig. 2. Average classifying layer processing times for the same point cloud on different computers: K1 – obstacle classifier, K2 – passability classifier

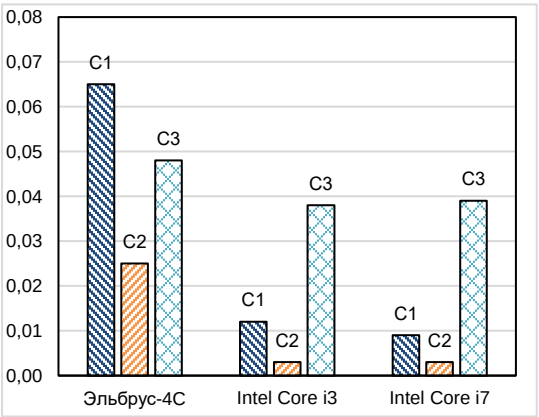

Рис. 3. Среднее время обновления слоев карты на различных ВК (в секундах): С1 – слой классификатора препятствий по данным 3D-лидара, С2 – слой классификатора профильной проходимости по данным 3D-лидара, С3 – слой классификатора препятствий по данным 2D-лидара

Fig. 3. Average processing time for map layers updates on different computers: C1 – obstacle classification layer for 3D LIDAR input, C2 – passability classification layer for 3D LIDAR input, C3 – obstacle classification layer for 2D LIDAR input

Пропорциональное снижение производительности наблюдается при обновлении слоев карты С1 и С2 (рис. 3). Предположительно, оно связано с накладными расходами на произвольный доступ к данным. Производительность при обновлении слоя С3 примерно одинакова для всех процессоров, это связано со временем ожидания преобразования систем координат от интегрированной в ROS библиотеки tf transform library [15].

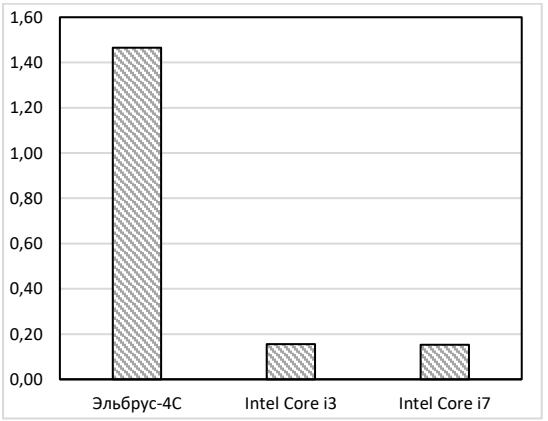

Рис. 4. Среднее время обновления сверточного слоя карты на различных ВК (в секундах)

Fig. 4. Average update time of the convolutive layer on various computers

Снижение производительности «Эльбрус-4С» наблюдается и при выполнении байесовской свертки слоев карты. При выполнении этой операции производительность снижается в 9,5 раза, что согласуется с результатами измерения производительности при работе классификаторов, а также с результатами сравнительного моделирования, приведенными в работе [4]. Следует также отметить, что во всех тестах на программно-аппаратной платформе «Эльбрус» наблюдалась полная работоспособность ROS.

Заключение

Проведенный анализ зависимостей, а также особенностей ПО платформы «Эльбрус» позволил успешно произвести портирование ROS и создать автоматизированные сценарии его сборки и развертывания. Тестирование ROS с помощью ПО системы управления мобильного РТК на задаче построения многослойной карты проходимости подтвердило полную работоспособность портированного фреймворка. Полученные данные по производительности «Эльбрус-4С» по сравнению с процессорами Intel полностью коррелируют с данными других авторов. Проигрыш «Эльбрус-4С» в произ-

водительности составляет 8–10 раз. По предварительным оценкам, при использовании «Эльбрус-8С» проигрыш составит не более 2–3 раз.

Для повышения производительности отдельных вычислительных узлов ROS необходимо использовать возможности платформы «Эльбрус» по многопоточной обработке данных, возможности библиотеки eml, а также применять дополнительную низкоуровневую оптимизацию программного кода.

Концепция ROS «один вычислительный узел – одна задача» в совокупности с гибким механизмом распределения вычислительных задач по сети и встроенным механизмом обмена данными позволяют легко декомпозировать сложные задачи управления робототехническими системами на подзадачи с их последующим разделением по имеющимся в распоряжении разработчика вычислительным ресурсам. Данное преимущество было продемонстрировано в разделе «Описание процедуры тестирования ROS на программно-аппаратной платформе «Эльбрус». Поэтому можно констатировать, что портирование ROS на программно-аппаратную платформу «Эльбрус» является существенным шагом в развитии отечественной робототехники в части формирования технического задела для создания перспективных образцов бортовых информационно-управляющих систем.

Литература

1. Корчак В.Ю., Лапшов В.С., Рубцов И.В. Перспективы развития наземных робототехнических комплексов военного и специального назначения // Изв. ЮФУ. Технич. науки. 2015. № 10. С. 83–95.

2. Антохин Е.А., Евтихов А.Н., Паничев В.А. Актуальные вопросы группового применения наземных робототехнических комплексов военного назначения // Робототехника и техническая кибернетика. 2019. Т. 7. № 1. С. 14–20. DOI: 10.31776/RTCJ.7102.

3. Kozhin A.S., Alfonso D.M., Klishin P.A., Slesarev M., Smirnov D.A., Kostenko V.O., Tikhorskiy V.V., Polyakov N.Yu., Demenko R.V., Kozhin E.S., Smirnova E.V., Smolyanov P.A., Gruzdov F.A., Sakhin Yu.Kh. The 5th Generation 28nm 8-Core VLIW Elbrus-8C Processor Architecture. Proc. Intern. Conf. EnT, 2016, pp. 86–90. DOI: 10.1109/ent.2016.027.

4. Бочаров Н.А., Парамонов Н.Б., Тимофеев Г.С., Панова О.Ю. Производительность вычислительной техники с процессором «Эльбрус-8С» на задачах робототехнического комплекса // Наноиндустрия. 2018. № 9. С. 79–84. DOI: 10.22184/1993-8578.2018.82.79.84.

5. Бочаров Н.А., Парамонов Н.Б., Славин О.А., Янко Д.В. Оценка перспектив использования вычислительных средств семейства «Эльбрус» при реализации алгоритмов распознавания в современных робототехнических комплексах // Вопросы радиоэлектроники. 2018. № 2. С. 99–105.

6. Mahtani A., Sanchez L., Fernandez E., Martinez A. Effective Robotics Programming with ROS. Packt Publ., UK, 2016, 515 p.

7. Горобец А.В., Нейман-заде М.И., Окунев С.К., Калякин А.А., Суков С.А. Производительность процессора «Эльбрус-8С» в суперкомпьютерных приложениях вычислительной газовой динамики. М.: Изд-во ИПМ им. М.В. Келдыша РАН, 2018. 20 с. DOI: 10.20948/prepr-2018-152.

8. Уорд Б. Внутреннее устройство Linux; [пер. с англ. М. Райтмана]. СПб: Питер, 2016. 384 с.

9. Стивенс У.Р., Раго С.А. UNIX. Профессиональное программирование. СПб: Питер, 2018. 944 с.

10. Шалаев М.А. Сборка компонентов программного обеспечения вычислительных комплексов семейства «Эльбрус» // Вопросы радиоэлектроники. 2017. № 3. С. 39–43.

11. Волосатова Т.М., Козов А.В., Рыжова Т.П. Анализ методов обнаружения препятствий системы навигации мобильного робота // Современное машиностроение. Наука и образование: матер. Междунар. науч.-практич. конф. СПб: Изд-во Политех. ун-та, 2018. № 7. С. 420–429. DOI: 10.1872/MMF-2018-36.

12. Kozov A.V., Volosatova T.M., Vukolov A.Y. Structural obstacle recognition method and its application in elevated terrain objects search. Proc. RusAutoCon, IEEE, 2018, pp. 1–5. DOI: 10.1109/RUSAUTOCON.2018.8501765.

13. Fankhauser P., Hutter M. A universal grid map library: implementation and use case for rough terrain navigation. The ROS Multimaster Extension for Simplified Deployment of Multi-Robot Systems. Springer, Cham, 2016, pp. 99–120. DOI: 10.1007/978-3-319-26054-9_5.

14. ROS RViz package homepage. URL: http://www.ros.org/wiki/rviz (дата обращения: 25.04.2019).

15. Foote T. Tf: The transform library. Proc. IEEE Conf. TePRA, 2013, pp. 1–6.

Software & Systems
DOI: 10.15827/0236-235X.128.655-664

Received 29.05.19
2019, vol. 32, no. 4, pp. 655–664

Peculiarities of porting the Robot Operating System framework onto Elbrus platform

A.A. Tachkov [1], *Ph.D. (Engineering), Head of the Department "Automated transport systems", tachkov@bmstu.ru*
A.Yu. Vukolov[1], *Programmer, twdragon@bmstu.ru*
A.V. Kozov [1], *Engineer, alexey.kozov@gmail.com*

[1] *Research and Training Center "Robotics" Bauman Moscow State Technical University, Moscow, 105037, Russian Federation*

Abstract. The Robot Operating System is the most widespread helper framework for mobile robots control systems development. However, it fully supports only Ubuntu/Debian Linux-based software, which leads to limitation of the possibility to use calculating equipment developed natively in Russia within design control systems. The authors ported ROS Melodic Morenia onto Elbrus platform natively developed in Russia (computer based on Elbrus-4C CPU).

This paper describes main peculiarities of the porting process associated with the differences between Elbrus operating system and most of Linux distributions. Version matching for Elbrus integrated software on which the ROS depends is also considered. The authors attempted to determine ROS build dependency libraries that could be fully installed on Elbrus using only system packages and to repack these libraries into deb-packages for further installation onto similar computers. In addition, the paper describes the developed deployment scenarios for ROS ready-out-of-the-box.

There is a description of testing of the prepared distribution performed for the task of multilayered passability map building. This task was solved on control system with LIDAR point cloud input. The authors demonstrate the results of tests as treatment times for the identical point clouds processing and map layer refreshment. The tests were run on systems based on Elbrus-4C, Intel Core i3-3220 and Intel Core i7-6700HQ CPUs using the same ROS version. The authors conclude that the ROS Melodic Morenia deployed onto Elbrus platform iы fully operational.

Keywords: Elbrus platform, ROS, porting, building software packages, performance testing, layered map, point cloud, mobile robot.

References

1. Korchak V.Yu., Lapshov V.S., Rubtsov I.V. Development prospects of military and special ground robots. *Izv. SFedU. Engineering Sciences*. 2015, no. 10, pp. 83–95 (in Russ.).

2. Antokhin E.A., Evtikhov A.N., Panichev V.A. Topical issues of group use of unmanned ground military robot. *Robotics and Technical Cybernetics*. 2019, vol. 7, no. 1, pp. 14–20. DOI: 10.31776/RTCJ.7102 (in Russ.).

3. Kozhin A.S., Alfonso D.M., Klishin P.A., Slesarev M., Smirnov D.A., Kostenko V.O., Tikhorskiy V.V., Polyakov N.Yu., Demenko R.V., Kozhin E.S., Smirnova E.V., Smolyanov P.A., Gruzdov F.A., Sakhin Yu.Kh. The 5th Generation 28nm 8-Core VLIW Elbrus-8C Processor Architecture. *Proc. Intern. Conf. on Engineering and Telecommunication (EnT)*. 2016, pp. 86–90. DOI: 10.1109/ent.2016.027.

4. Bocharov N.A., Paramonov N.B., Timofeev G.S., Panova O.Yu. Perfomance of computer systems with Elbrus-8S processor for robotic systems tasks. *Nanoindustriya*. 2018, no. 9, pp. 79–84. DOI: 10.22184/1993-8578.2018.82.79.84 (in Russ.).

5. Bocharov N.A., Paramonov N.B., Slavin O.A., Yanko D.V. Assessment of perspectives of exploitation of computer systems Elbrus for implementing recognition algorithms in modern robotic complexes. *Issues of Radio Electronics*. 2018, no. 2, pp. 99–105 (in Russ.).

6. Mahtani A., Sanchez L., Fernandez E., Martinez A. *Effective Robotics Programming with ROS*. 3rd ed., Packt Publ., 2016, 556 p.

7. Gorobets A.V., Nejman-zade M.I., Okunev S.K., Kalyakin A.A., Sukov S.A. *Elbrus-8C Processor Performance in Supercomputer Applications of Computational Gas Dynamics*. 2018, no. 152, 20 p. DOI: 10.20948/prepr-2018-152.

8. Ward B. *How Linux Works: What Every Superuser Should Know*. 2nd ed., 2014, 394 p. (Russ. ed.: M. Raytman, St. Petersburg, Piter Publ., 2016, 384 p.).

9. Stivens W.R., Rago S.A. *UNIX. Advanced Programming in the UNIX Enviroment*. 3rd ed. Addison-Wesley Prof. Publ., 2013, 1032 p. (Russ. ed.: St. Petersburg, Piter Publ., 2018, 944 p.).

10. Shalaev M.A. Building software for Elbrus microprocessor family. *Issues of Radio Electronics*. 2017, no. 3, pp. 39–43 (in Russ.).

11. Volosatova T.M., Kozov A.V., Ryzhova T.P. Analysis of methods for detecting obstacles in the navigation system of a mobile robot. *Proc. 7th Intern. Sci. and Pract. Conf. "Modern engineering. Science and education" (MMESE-2018)*. St. Petersburg, 2018, pp. 420–429. DOI: 10.1872/MMF-2018-36 (in Russ.).

12. Kozov A.V., Volosatova T.M., Vukolov A.Y. Structural obstacle recognition method and its application in elevated terrain objects search. *Proc. 2018 Intern. Russ. Automation Conf. (RusAutoCon), IEEE*. 2018, pp. 1–5. DOI: 10.1109/RUSAUTOCON.2018.8501765.

13. Fankhauser P., Hutter M. A Universal grid map library: implementation and use case for rough terrain navigation. *The ROS Multimaster Extension for Simplified Deployment of Multi-Robot Systems*. Springer, Cham, 2016, pp. 99–120. DOI: 10.1007/978-3-319-26054-9_5.

14. *ROS RViz*. Available at: http://www.ros.org/wiki/rviz (accessed April 25, 2019).

15. Foote T. *Tf: The transform library. IEEE Conf. Technologies for Practical Robot Applications (TePRA)*. 2013, pp. 1–6.

Для цитирования

Тачков А.А,. Вуколов А.Ю., Козов А.В. Особенности портирования Robot Operating System на программно-аппаратную платформу семейства «Эльбрус» // Программные продукты и системы. 2019. Т. 32. № 4. С. 655–664. DOI: 10.15827/0236-235X.128.655-664.

For citation

Tachkov A.A., Vukolov A.Yu., Kozov A.V. Peculiarities of porting the Robot Operating System framework onto Elbrus platform. *Software & Systems*. 2019, vol. 32, no. 4, pp. 655–664 (in Russ.). DOI: 10.15827/0236-235X.128.655-664.