Новосибирский государственный университет
Факультет Информационных Технологий
Кафедра Параллельных Вычислений

ОТЛАДКА ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

Власенко Андрей Юрьевич

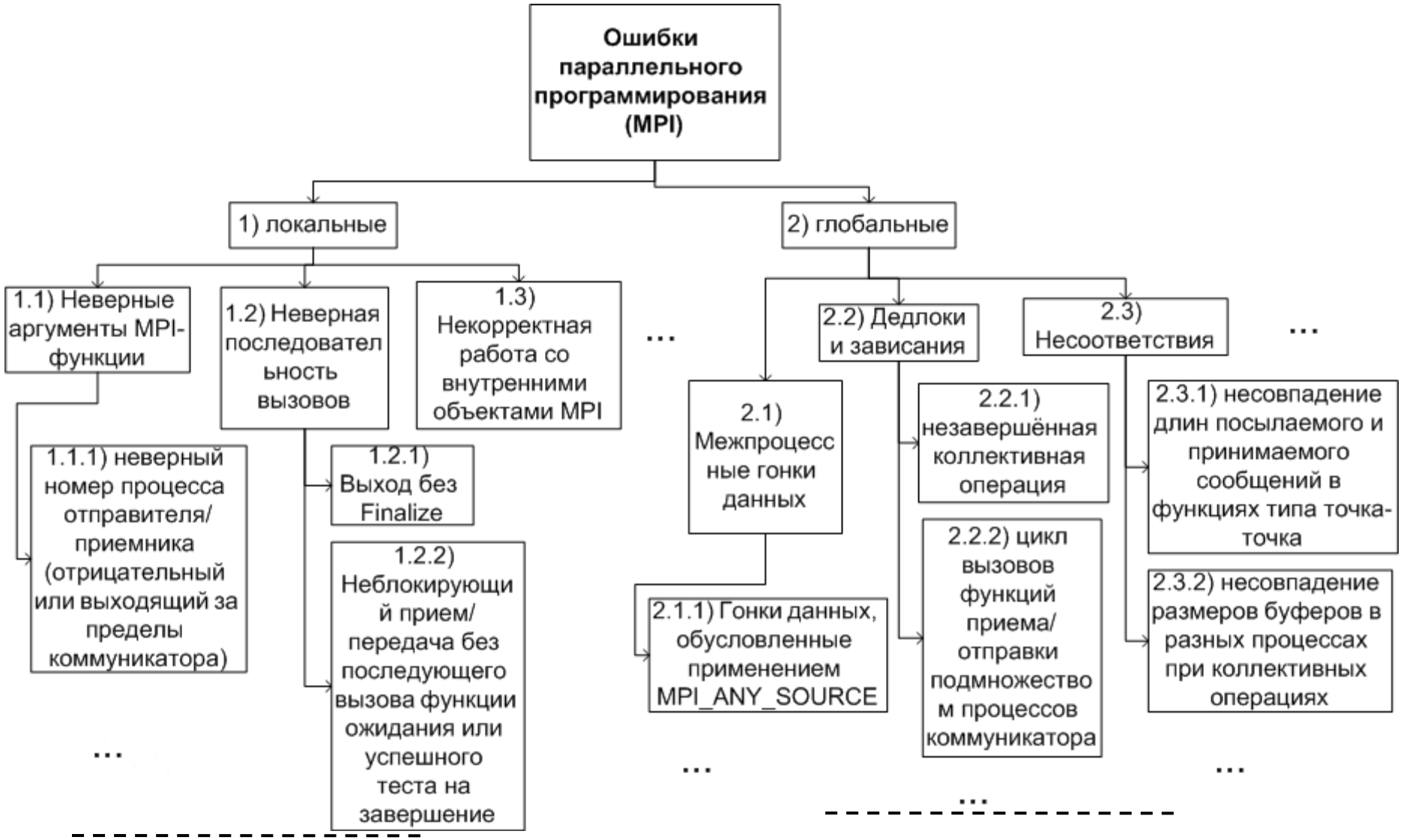
Проблема отладки параллельных программ

- Вместе с галопирующим экстенсивным ростом производительности суперкомпьютеров увеличиваются и масштабы параллельных программ.
- Обнаружение логических ошибок в программах из сотен и тысяч процессов и потоков – крайне сложная задача.
- Наряду с ошибками, свойственными последовательным программам, параллельные программы могут содержать множество ошибок новых типов.

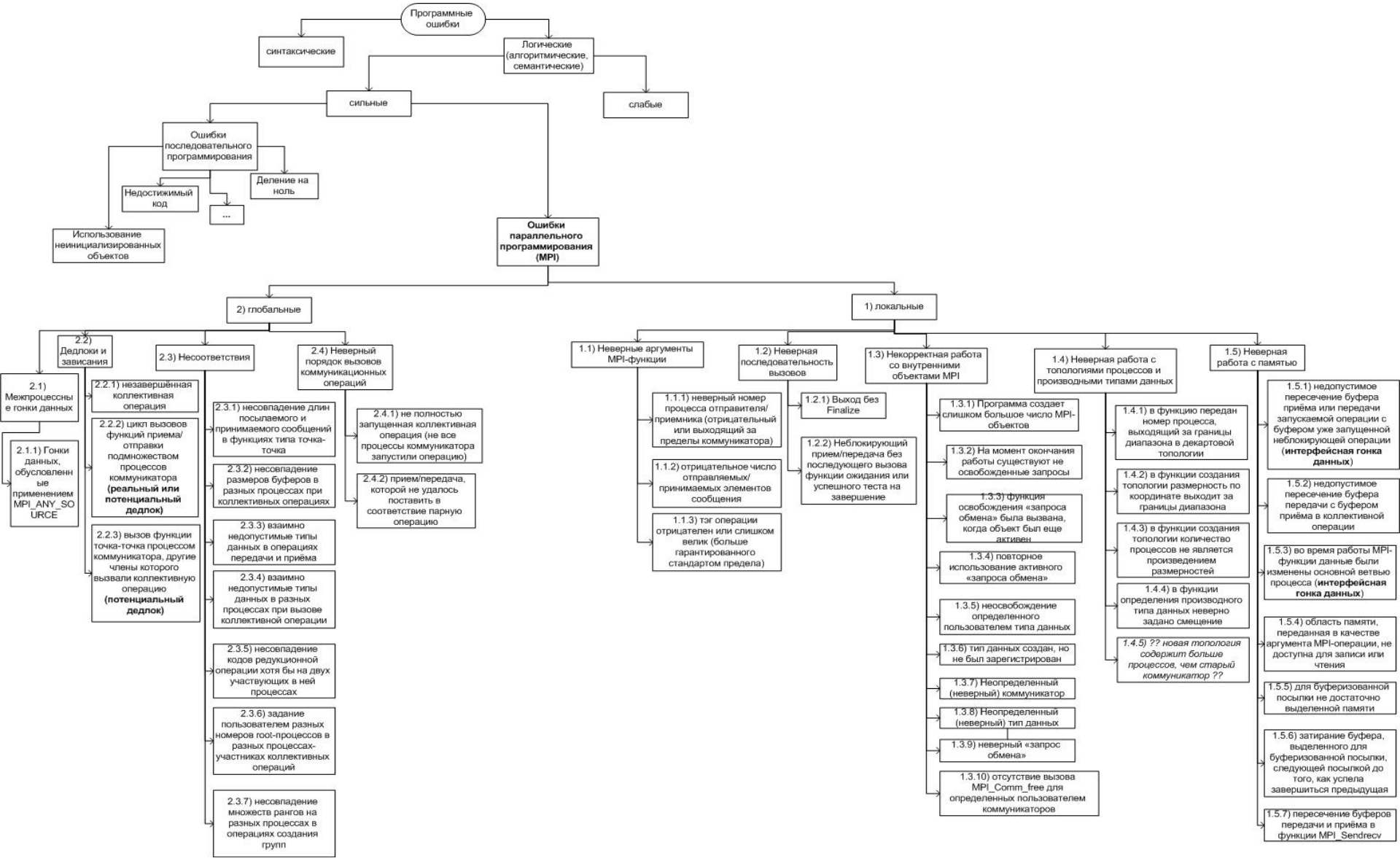
Программные ошибки

1) К.Н. Ефимкин, О.Ф. Жукова, В.А. Крюков «Средства отладки MPI-программ. Анализатор корректности»
2) B.Kramer, M.Müller «Runtime Checking of MPI Applications using MARMOT»
3) Intel Trace Collector reference guide

Ошибки в программах MPI-1



Ошибки в программах MPI-1



Подходы к отладке параллельных программ

1. Диалоговая отладка (TotalView, Distributed Debugging Tool)
2. Статический анализ
3. Верификация программы на модели (Spin, MPI-Spin)
4. Сравнительная отладка (Distributed Virtual Machine - DVM)
5. Автоматизированный контроль корректности*)
   a. во время исполнения программы (ITAC,Marmot,MUST)
   b. по трассе (DVM, Intel Message Checker)

*) Ефимкин, К. Н. Средства отладки MPI-программ. Анализатор корректности [Электронный ресурс]/ К. Н. Ефимкин, О. Ф. Жукова, В.А.Крюков// http://www.keldysh.ru/papers/2006/prep28/prep2006_28.html

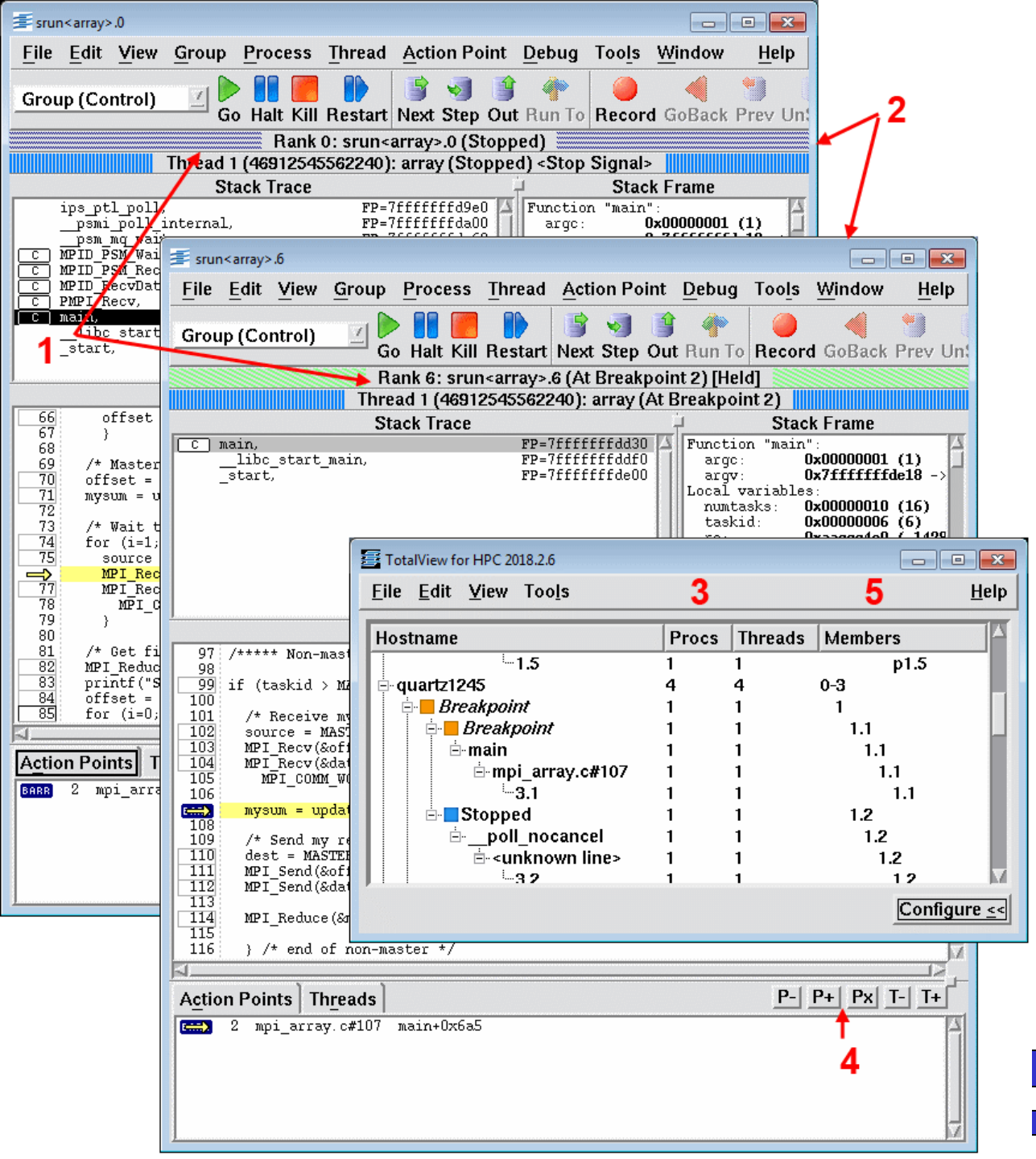
Диалоговая отладка

Наиболее популярны диалоговые отладчики (TotalView, Allinea DDT).
Минусы диалоговых отладчиков:
•При запуске множества процессов крайне сложно отслеживать поведение, двигаясь по шагам в каждом из процессов.

•Диалоговые отладчики подразумевают интерактивную работу с приложением, что не всегда возможно в условиях суперкомпьютерного центра.

Вывод: необходимы инструментальные средства, использующие **автоматизированные** методы отладки

Вид TotalView



Статический анализ

Статический анализ кода (англ. static code analysis) - анализ программного обеспечения, производимый по исходному коду без реального выполнения исследуемых программ [1].

Преимущества: [2]

- Полное покрытие кода.
- Результат анализа не зависит от используемого компилятора.
- Результат анализа не зависит от используемой реализации MPI.

---

1. Марков А.С., Фадин А.А. Статический анализ безопасности кода [Электронный ресурс] // Программная инженерия и информационная безопасность. URL: https://npo-echelon.ru/doc/stat-analisys.pdf (Дата обращения: 14.03.2017)
2. Статический анализ кода [Электронный ресурс] // URL: https://www.viva64.com/ru/t/0046/ (Дата обращения: 15.03.2017)

Популярные статические анализаторы

- CppCheck

- Clang Static Analyzer (LLVM)

- Coverity Scan

- PVS-Studio

Недостаток:

Эти анализаторы не определяют ошибки, специфичные для MPI-программ!

Сравнительная отладка

Принцип – сравнительный анализ выполнения эталонной и анализируемой версий программы.

Примеры систем: DVM, Guard

Проблемы:

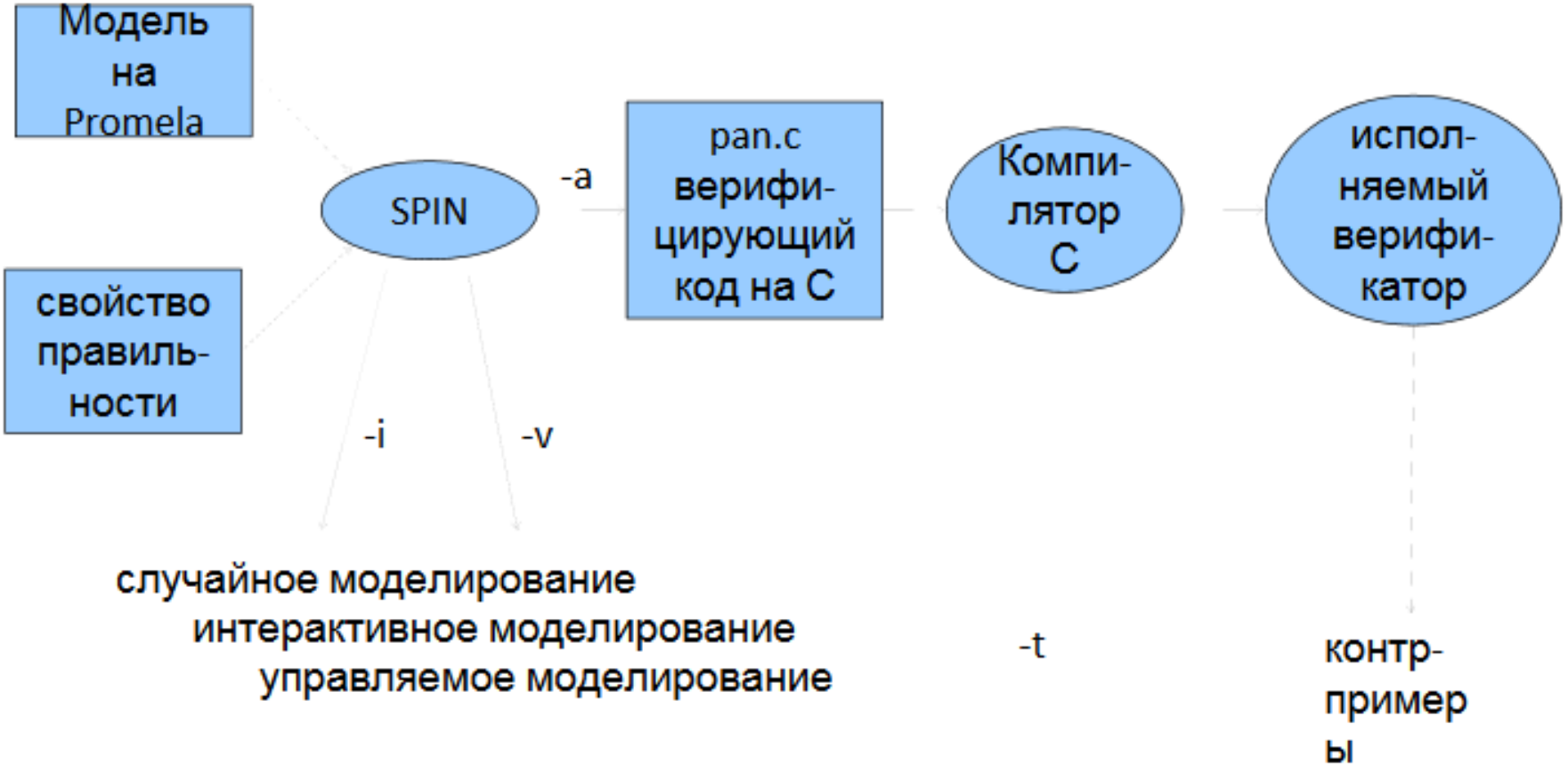
- Эталонная версия существует далеко не всегда.
- Многие ошибки, обусловленные взаимодействием параллельных ветвей приложения, не обнаруживаются сравнением с последовательной версией.

Верификация на модели

Принцип – проверка модели программы, составленной на некотором псевдоязыке, специфицированным требованиям. Модель – система состояний программы и переходов между состояниями (граф).

Примеры систем: Spin, MPISpin

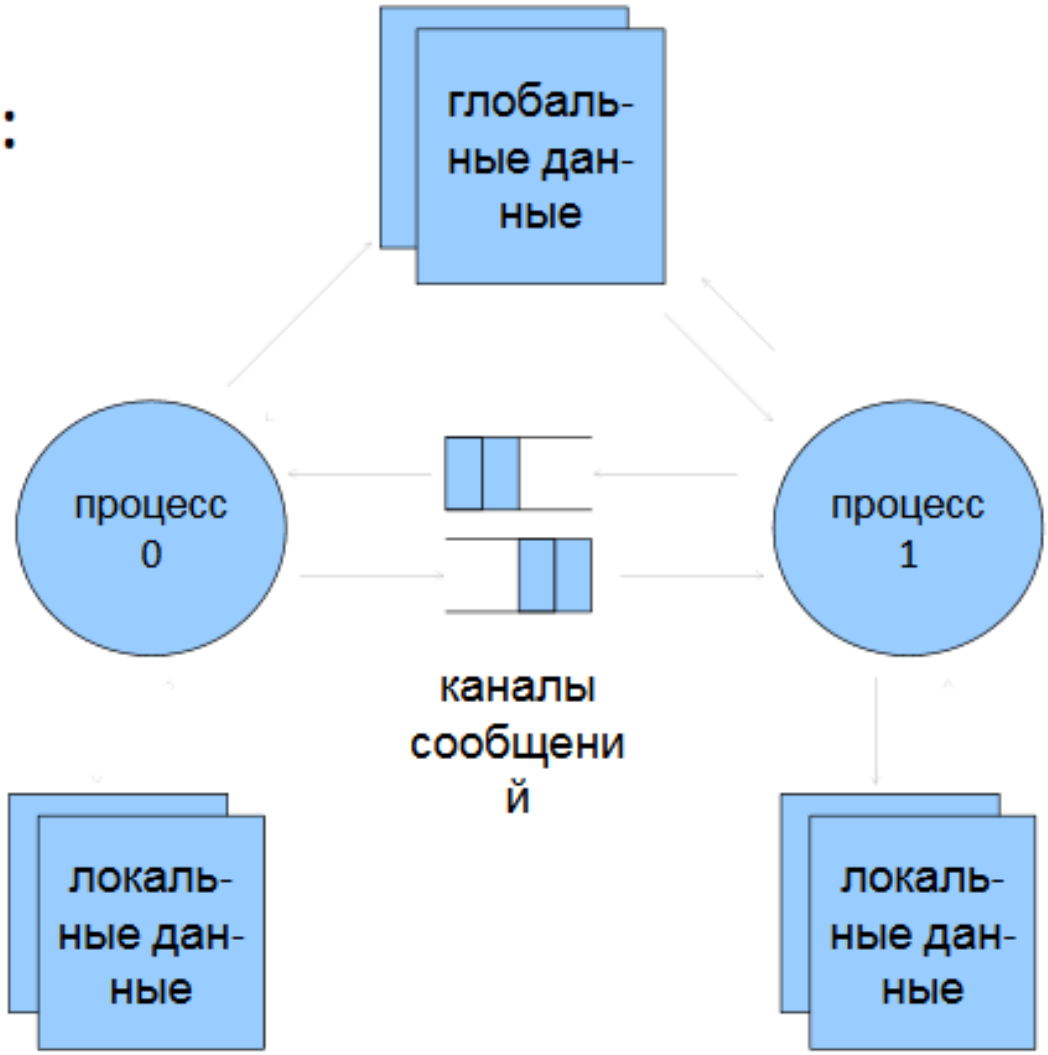
Процесс верификации



Устройство модели на Promela

Три типа объектов:

- процессы,
- глобальные и локальные объекты данных,
- каналы сообщений.



Проблемы верификации на модели

1) Не подходит для обнаружения многих ошибок, обусловленных неверным использованием стандарта MPI. В первую очередь – большинства локальных ошибок.

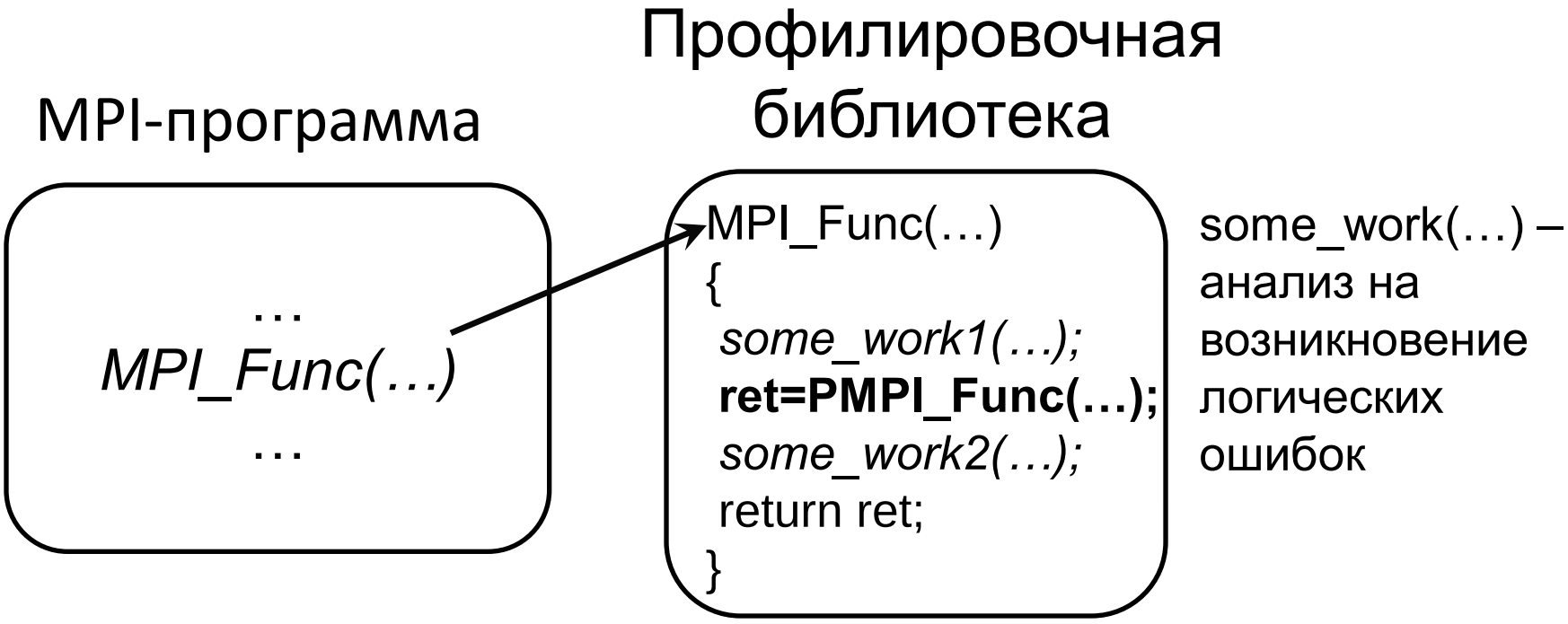
2) «Комбинаторный взрыв» - экспоненциальный рост количества состояний от числа процессов параллельной программы

Автоматизированный контроль корректности

- Принцип – проверка исполнения программы на соответствие некоторым правилам (для MPI - правилам MPI-стандарта).

- Примеры систем: MUST, MARMOT, Intel Trace Analyzer & Collector

Автоматизированный контроль корректности MPI-программ



Плюсы автоматизированного контроля корректности во время исполнения

- Обнаруживает ошибки, обусловленные недетерминизмом
  ⟷ *диалоговая отладка*
- Нет необходимости создавать эталонную версию
  ⟷ *сравнительная отладка*
- Нет необходимости создавать модель программы на псевдоязыке ⟷ *верификация программы на модели*
- Нет неконтролируемо увеличивающегося файла трассы
  ⟷ *анализ по трассе*

Проблемы автоматизированного контроля корректности

- Большое количество возможных логических ошибок, для каждой необходима своя проверка.
- Регулярное обновление MPI-стандарта (версии 1.0, 1.1, 1.2, 1.3, 2.0, 2.1, 2.2, 3.0) => новые функции и объекты => новые ограничения => новые возможные ошибки
- Невозможность осуществлять пользовательские проверки.
- Системы собирают информацию о пользовательской программе при помощи профилировочного интерфейса MPI, поэтому область применимости ограничивается неверным использованием функций и объектов MPI.

Научная работа:

АВТОМАТИЗИРОВАННЫЙ КОНТРОЛЬ КОРРЕКТНОСТИ MPI-ПРОГРАММ НА ОСНОВЕ ШАБЛОНОВ ОШИБОЧНОГО ПОВЕДЕНИЯ

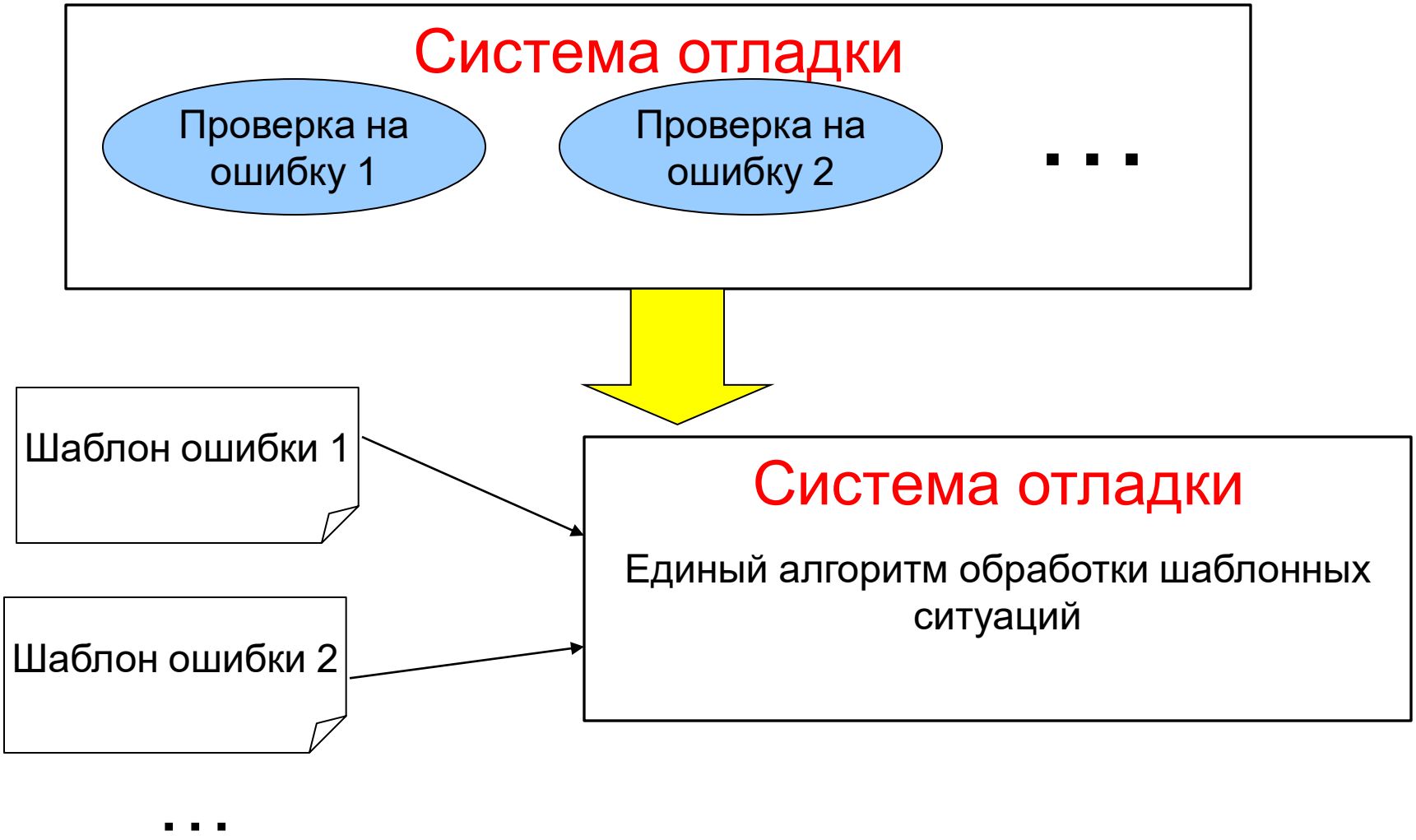
Цель работы

Разработка и реализация эффективного метода отладки параллельных программ, базирующегося на принципах автоматизированного контроля корректности с использованием текстовых шаблонов ошибочного поведения.

Задачи исследования

1. Выполнить обзор и анализ методов и инструментальных средств отладки параллельных программ.
2. Разработать систему описания шаблонов ошибочного поведения параллельных программ, использующих интерфейс MPI.
3. С помощью разработанной системы описания формализовать наиболее распространенные семантические ошибки в MPI-приложениях.
4. Создать инструментальное средство отладки, выполняющее поиск по заранее составленным текстовым шаблонам ошибочных ситуаций, возникающих при исполнении MPI-приложений.
5. Провести тестирование инструментального средства отладки и сопоставить его с аналогами.

Механизм текстовых шаблонов – основа метода

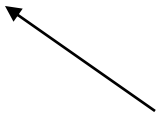


Система описания шаблонов ошибочного поведения

1 ошибочная ситуация -> 1 файл шаблона

Структура файла шаблона:

1) Название шаблона
2) Блок описания процессов (количество, [номера])
3) Блок описания функций
4) Блок условий на аргументы функций



Параметры, которые можно получить при помощи профилировочного интерфейса MPI

Пример: шаблон «неосвобождение определенного пользователем типа данных на нулевом процессе»

Name=Unfreed user defined datatype
PROCESSES
K=1 p1=0
FUNCTIONS
F1=p1:Type_contiguous
F2=p1:Type_free!
CONDITIONS
F1(3)=F2(1)

Определение системы в расширенной форме Бэкуса-Наура

шаблон ::= ‘Name’, ‘=’, строка_символов,
‘PROCESSES’,
‘K’, ‘=’, количество_процессов [, номера_процессов],
‘FUNCTIONS’,
{‘F’, натуральное_число, ‘=’, ‘p’, натуральное_число, ‘:’, функция [{, ‘||’, функция}] [{, ‘&&’, функция}] },
‘CONDITIONS’,
{(‘F’, натуральное_число, ‘(’, натуральное_число, ‘)’, (‘=’ | ‘!=’ | ‘>’ | ‘<’), значение_аргумента)
| (‘CRC_BEFORE’, ‘(’, ‘F’, натуральное_число, ‘(’, натуральное_число, ‘)’, ‘)’, ‘!=’, ‘CRC_AFTER’, ‘(’, ‘F’, натуральное_число, ‘(’, натуральное_число, ‘)’, ‘)’)}
;
строка_символов ::= {символ};
количество_процессов ::= натуральное_число | ‘n(MPI_COMM_WORLD)’;
номера_процессов ::= {‘ ’, ‘p’, натуральное_число, ‘=’, натуральное_число};
функция ::= макрос | имя_MPI_функции_без_префикса_MPI [, ‘!’];
…

Плюсы введения механизма шаблонов

- Простота добавления новых ситуаций для обнаружения.
- Возможность конкретизации параметров (например, указываем конкретные номера процессов и для других процессов не производим проверки).
- Возможность отключать какие-либо шаблоны.
- Возможность задавать пользовательские проверки на какие-либо аргументы (например, пользователь может задать, что аргумент №2 функции Send должен быть в пределах 1<=F(2)<50).

Составленные шаблоны

22 семантические ошибки (20 – для функций стандарта MPI-1 и 2 – для MPI-2) и 2 пользовательских проверки

1. Несовпадение длин посылаемого и принимаемого сообщений в функциях точка-точка.
2. Взаимно недопустимые типы данных в операциях передачи и приема.
3. Гонки данных, обусловленные применением MPI_ANY_SOURCE.
4. Дедлоки, обусловленные циклом отправок или получений.
5. Вызов коммуникации точка-точка процессом, парный для которого вызвал коллективную операцию.

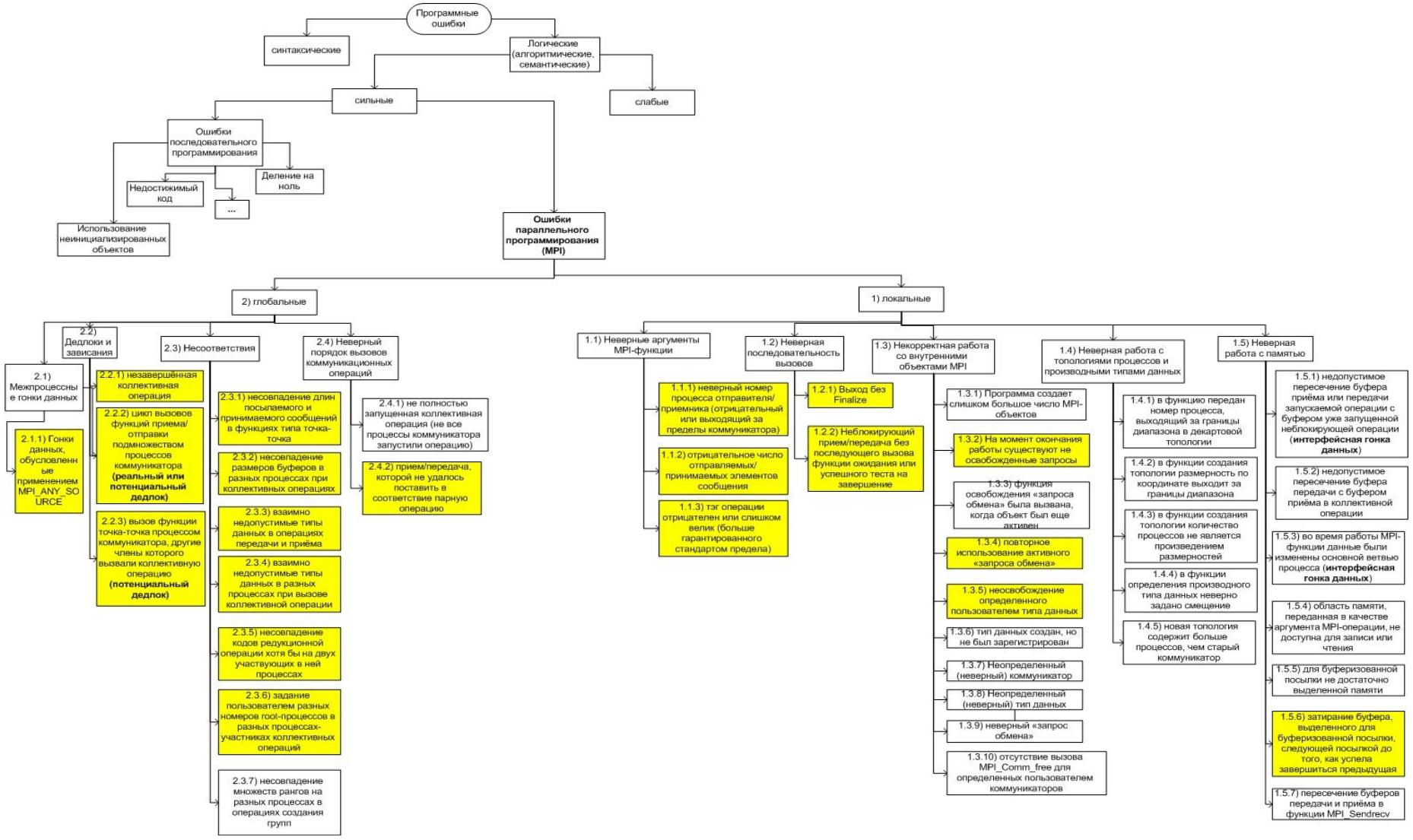
…

21. Неверный режим доступа при открытии файла *(MPI-2)*.

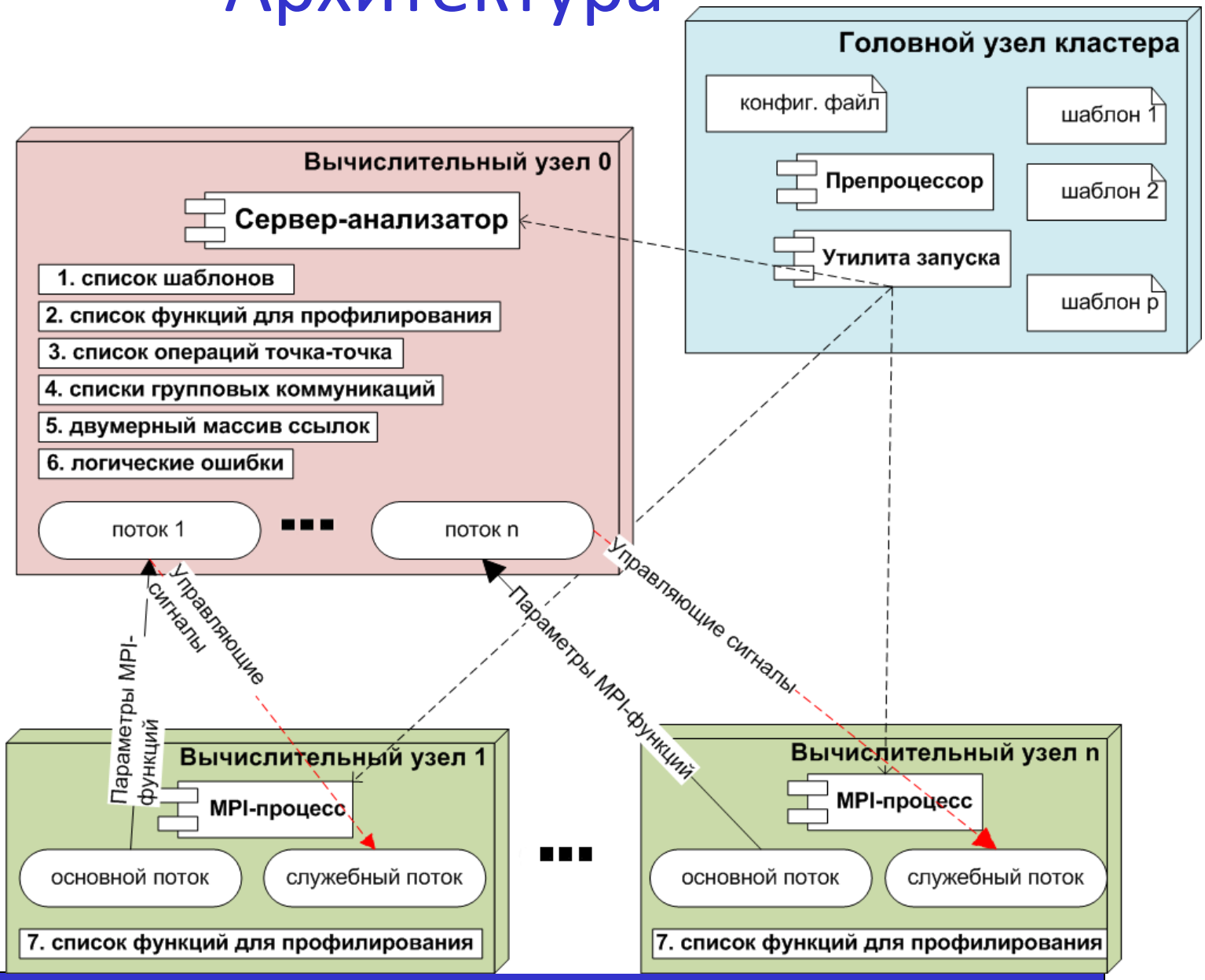
…

24. Коммуникации между процессами, разность номеров которых более единицы при пересылке данных по кольцу *(пользоват.)*

«Покрытие» ошибок шаблонами



Архитектура



Конфигурационный файл

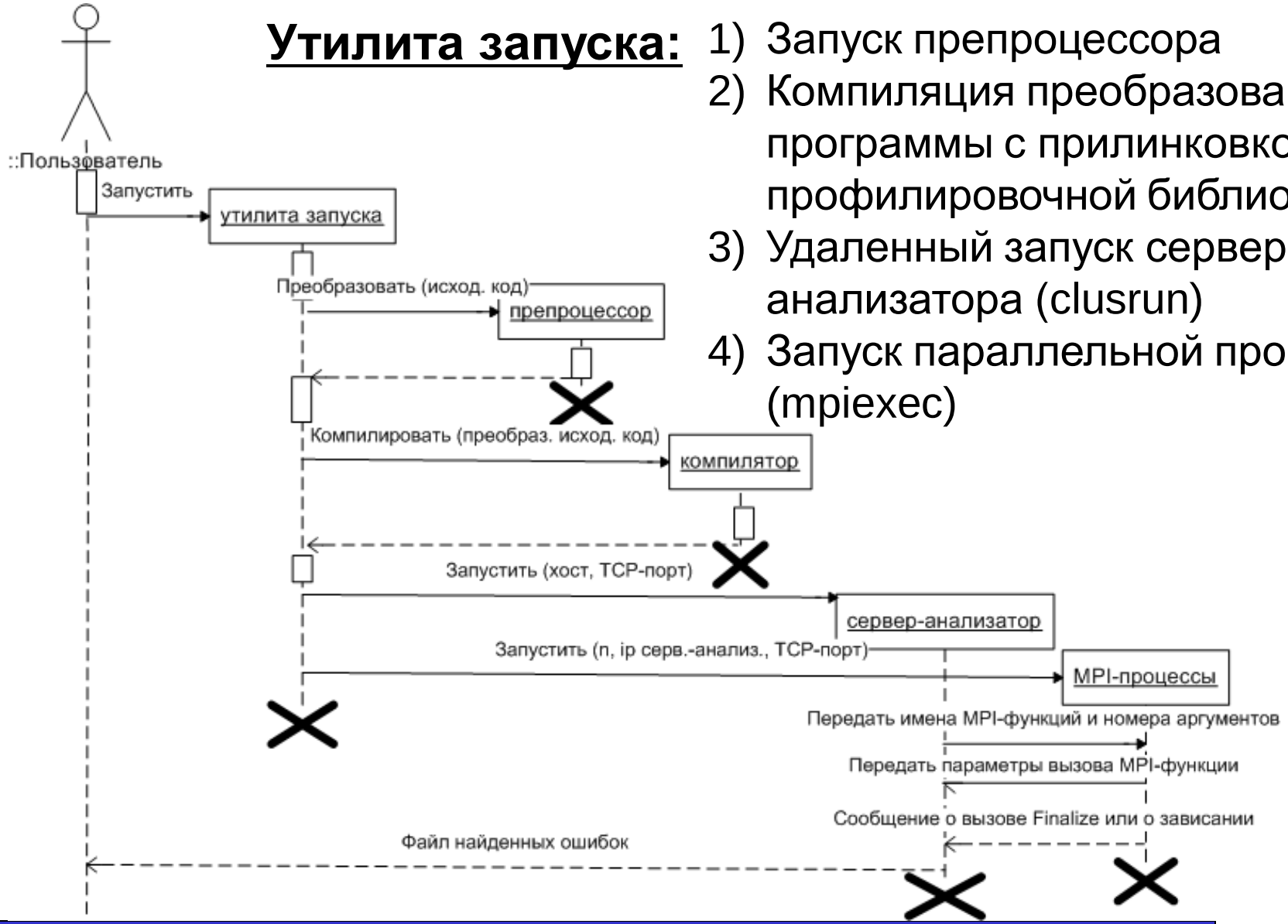
- имена файлов исходного кода;
- путь до директории с файлами шаблонов;
- количество MPI-процессов
- ip-адрес (имя) сервера-анализатора
- путь к библиотеке MPICH
- путь к препроцессору
- путь к профилировочной библиотеке
- рабочая директория проекта (папка общего доступа)
- [имена вычислительных узлов]
- [TCP-порты для обмена данными между сервером и MPI-процессами]
- [путь к компилятору]
- [имя файла ошибок]

[*абв*] – необязательные параметры

Алгоритм работы программного средства

Утилита запуска:

1) Запуск препроцессора
2) Компиляция преобразованной программы с прилинковкой профилировочной библиотеки
3) Удаленный запуск сервера-анализатора (clusrun)
4) Запуск параллельной программы (mpiexec)



Содержание выходного файла

– Имя шаблона
– Для каждой функции, вызвавшей срабатывание шаблона:
  - имя функции
  - имя файла исходного кода
  - номер строки исходного кода
  - номер MPI-процесса (в коммуникаторе MPI_COMM_WORLD)
  - Параметры функции:
    - [тип данных]
    - [количество элементов]
    - [номер парного процесса]
    - [тэг сообщения]
    - [Root-процесс]

БЛАГОДАРЮ ЗА ВНИМАНИЕ!