SCALABILITY:
вопросы масштабируемости

Dmitry Mishura

Agenda – или о чем речь

1. Кластерные вычисления
2. MPI как промышленный стандарт
3. Масштабируемость как метрика качества ПО
4. «Грабли» параллельных программ
5. IMC: контроль корректности MPI вызовов
6. ITAC: сбор и анализ статистики MPI
7. Документация и ссылки

Кластерные вычисления

1. «Неограниченная» вычислительная мощность
2. Независимые вычислительные узлы
3. Процесс как контейнер вычислений
4. Распределенная память
5. Общая файловая система
6. Быстрая сеть
7. Очередь задач

MPI как промышленный стандарт

1. Существует единый стандарт MPI
2. Как платные так и бесплатные реализации
3. Широкая переносимость между платформами
4. Поддержка гетерогенных кластеров
5. Отличная поддержка аппаратных возможностей (сети, FS)

Intel MPI 4.0: скорость прежде всего

1. Самые быстрые механизмы внутри-узлового обмена
2. Большой спектр поддержки современных сетей: Ethernet, IB (DAPL,OFA), TMI
3. Поддержка нескольких сетевых адаптеров (Multirail)
4. Быстрое порождение процессов (HYDRA)
5. Оптимизированная библиотека ADIO
6. Оптимизированные коллективные операции: как алгоритмически так и архитектурно
7. Оптимизированная очередь сообщений

Дополнительные возможности

1. Fault tolerance – устойчивость к сбоям сети
2. «Умная» привязка процессов
3. Динамическое создание соединений (DAPL)
4. Интеграция с библиотеками и утилитами Intel

Масштабируемость как метрика качества ПО

Масшта́бируемость (scalability) — в информатике означает способность системы увеличивать свою производительность при добавлении ресурсов (обычно аппаратных)

Программа называется *масштабируемой*, если она способна увеличивать производительность пропорционально дополнительным вычислительным мощностям

Strong scale – фиксированная задача, изменение производительности ПО при добавлении узлов

Weak scale – изменение производительности при добавлении узлов вместе с пропорциональным увеличением задачи (объема данных)

Плохо масштабируемое ПО не сможет использовать все возможности кластерной вычислительной системы

Принципиальные ограничения на масштабируемость

Следствие закона Амдала – если ваше ПО имеет нераспараллеленные участки, то масштабируемость будет хуже линейной (<1)

Принципиально не распараллеливаются:

1. Алгоритмические автоматы с внутренним состоянием;
2. Ввод-вывод
3. ?

Физические ограничения на минимальную латентность сети

«Грабли» параллельных программ или почему ПО не масштабируется

Пример из жизни N1 (МГУ):

```
…
for (int i=0;i<arraySize;i++){
        MPI_Send(array[i], msglen, MPI_DOUBLE, dest, tag,
                                MPI_COMM_WORLD);
}
...
```

«Грабли» параллельных программ или почему ПО не масштабируется

Пример из жизни N2 (МГУ):

```
...
if (isRoot)
for (int i=0;i<TotalRank;i++){
        int dest = i;
        MPI_Send(pMsg, msglen, MPI_CHAR, dest, tag,
        MPI_COMM_WORLD);
} // for
}else{
        MPI_Recv(pMsg, msglen, MPI_CHAR, root, tag,
        MPI_COMM_WORLD);
} //if
...
```

«Грабли» параллельных программ или почему ПО не масштабируется

Пример из жизни N3 (Open Source, Mocassine (астрофизика):

```
…
DoSomeWork();
MPI_Barrier(MPI_COMM_WORLD);

DoAnotherWork();

MPI_Barrier(MPI_COMM_WORLD);
MPI_Alltoall(..);
MPI_Barrier(MPI_COMM_WORLD);
…
```

«Грабли» параллельных программ или почему ПО не масштабируется

Пример из жизни N4 (Open Source):

```
…
for (int step=0; step<100000; step++){
        results = DoSomeWork();
        printf(“Step %i intermediate result\n”);
}
…
```

Intel Message Checker: модель использования

У вас уже есть параллельная программа

1. Прописываем путь к IMC: export VT_ROOT=/opt/intel/itac
2. Компилируем программу: mpiicc **–check** –O3 –xSSE4.2 main.c
3. Запускаем программу как обычно: mpirun –np 100500 ./main.bin

Intel Message Checker: пример использования

[0] WARNING: LOCAL:MEMORY:OVERLAP: warning
[0] WARNING**: New send buffer overlaps with currently active send buffer at address** 0x7fbfffea24.
[0] WARNING: Control over active buffer was transferred to MPI at:
[0] WARNING: MPI_Isend(*buf=0x7fbfffea24, count=1, datatype=MPI_CHAR, dest=1, tag=100,\
comm=MPI_COMM_WORLD, *request=0xaefca0)
[0] WARNING: main (local/memory/overlap/MPI_Isend.c:62)
[0] WARNING: Control over new buffer is about to be transferred to MPI at:
[0] WARNING: MPI_Isend(*buf=0x7fbfffea24, count=1, datatype=MPI_CHAR, dest=2, tag=100,\
comm=MPI_COMM_WORLD, *request=0xaefca4)
[0] WARNING: main (local/memory/overlap/MPI_Isend.c:62)

Intel Message Checker: какие ошибки можно поймать

1. Несоответствие размеров буферов и типов данных
2. Переиспользование буфера сообщения до завершения операции
3. «Потерянные» сообщения
4. Неосвобожденные после передачи ресурсы
5. Различные дедлоки

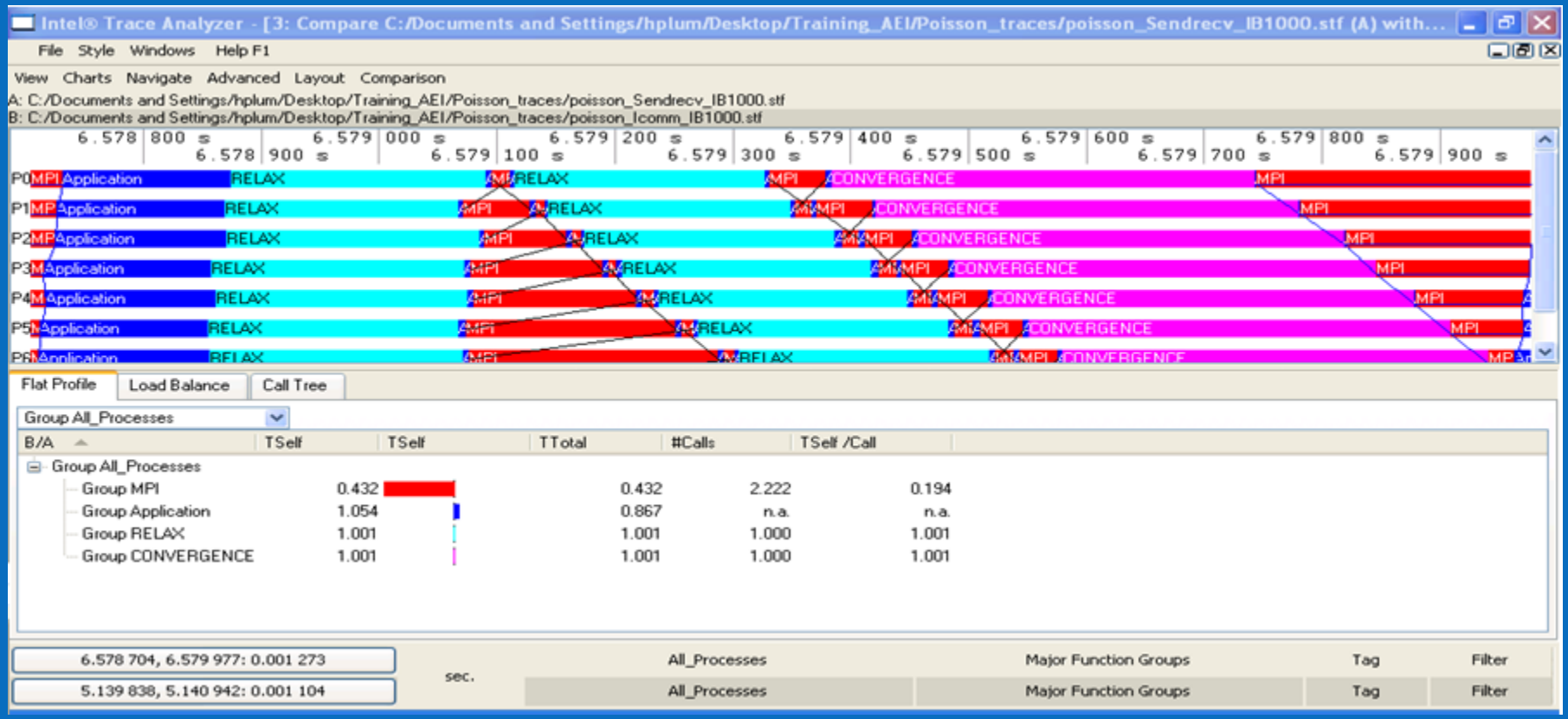
Intel Trace Collector: модель использования

У вас уже есть параллельная программа

1. Прописываем путь к ITAC: export VT_ROOT=/opt/intel/itac
2. Компилируем программу: mpiicc **–mpi_trace** –O3 –xSSE4.2 main.c
3. Запускаем программу как обычно: mpirun –np 100500 ./main.bin
4. <собирается трасса>
5. Запускаем Intel Trace Analyzer (GUI), анализируем полученную трассу

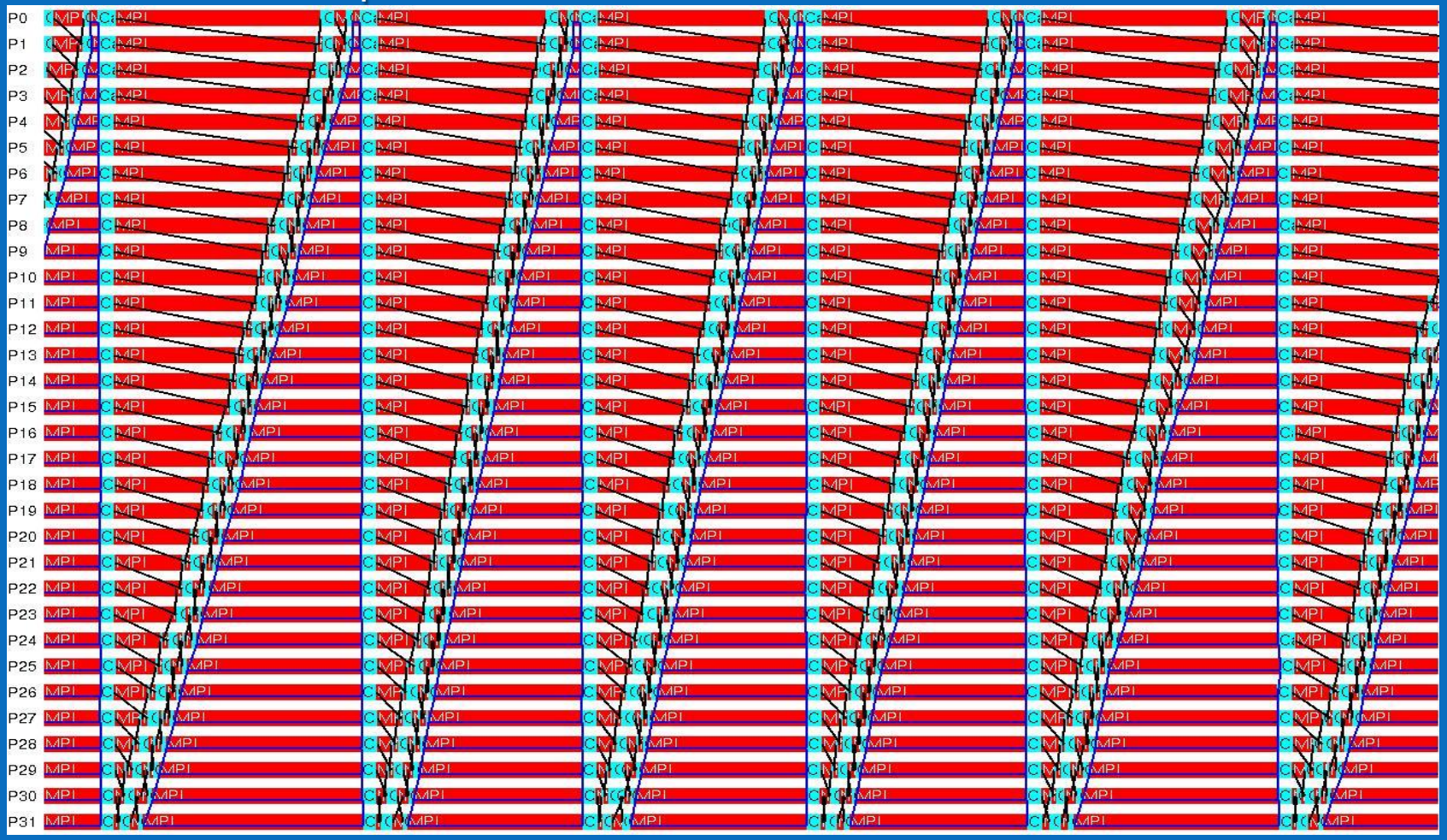
Intel Trace Analyzer: пример использования

Timeline: overview



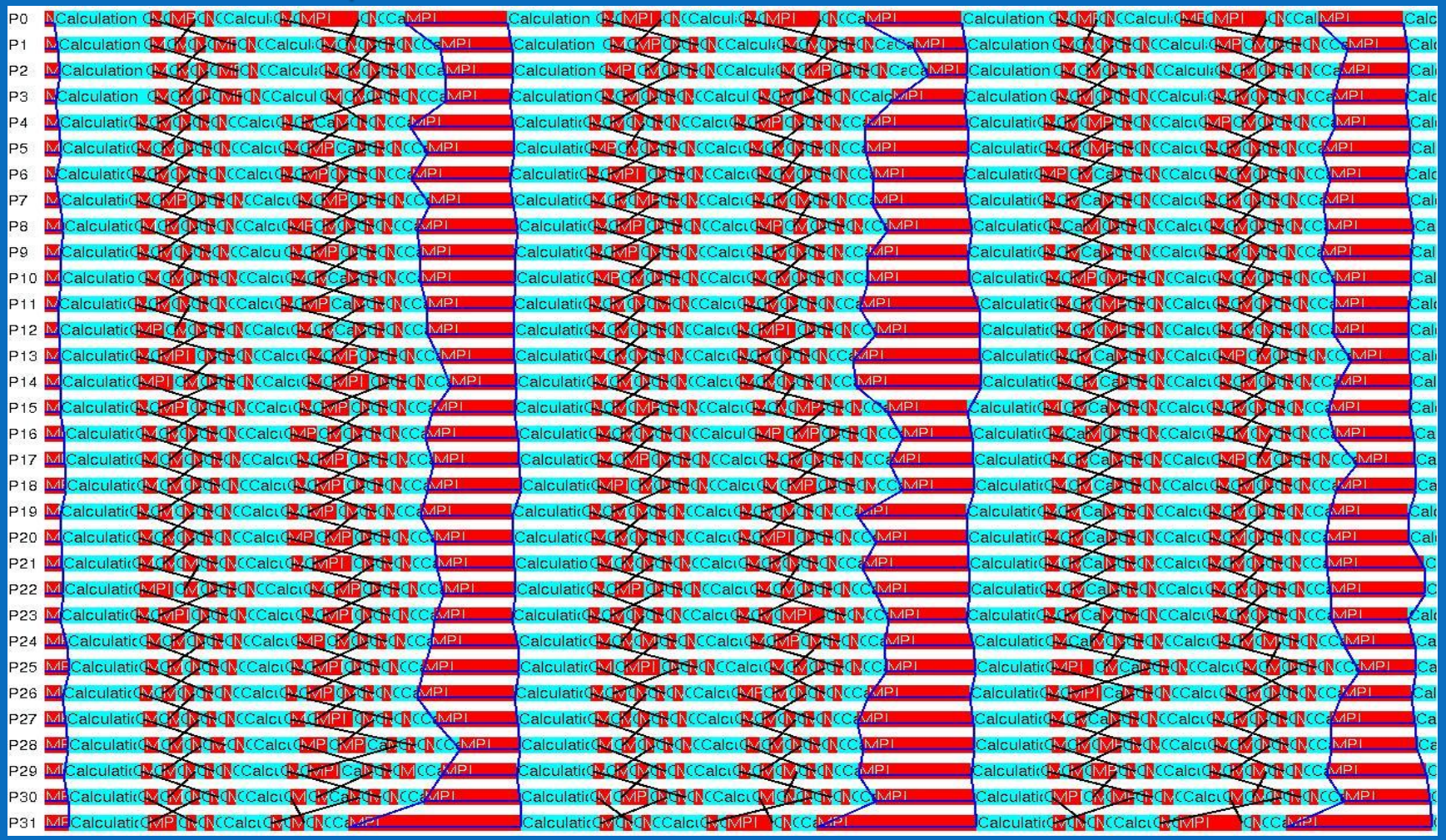
Intel Trace Analyzer: пример использования

Timeline: bad example



Intel Trace Analyzer: пример использования

Timeline: bad example

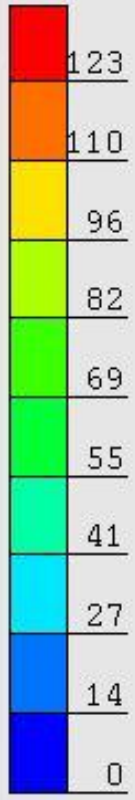


Intel Trace Analyzer: пример использования

Профиль: коллективные операции

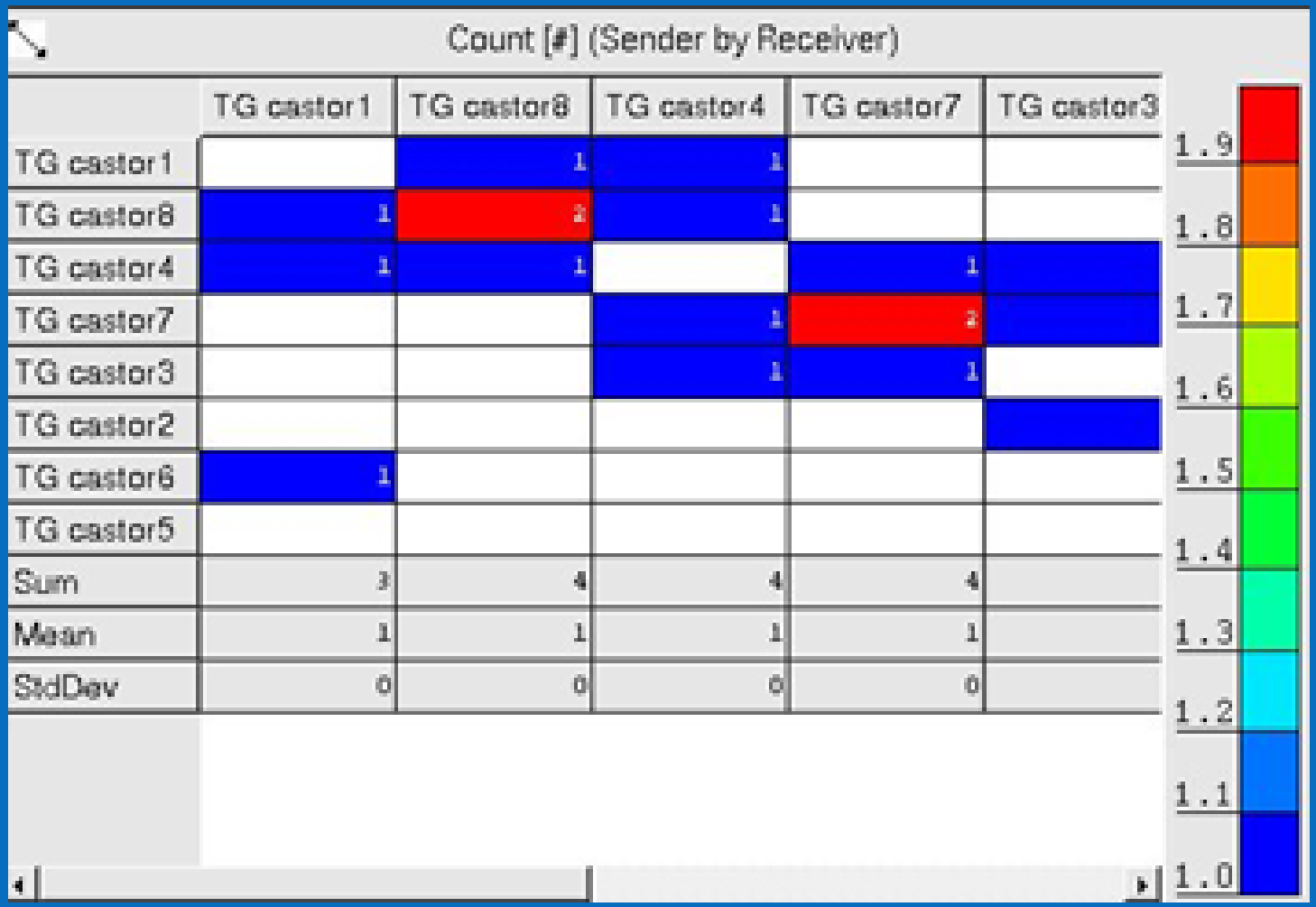
Total Time [s] (Collective Operation by Process)

	P0	P1	P2	P3	P4	P5	P6	P7	Sum	Mean	StdDev
MPI_Barrier	0.063	0.052	0.040	0.180	0.258	0.066	0.079	0.215	0.952	0.119	0.080
MPI_Bcast	0.000	0.860	0.865	0.857	0.853	0.855	0.860	0.861	6.010	0.751	0.284
MPI_Allreduce	87.299	120.679	88.085	127.782	89.071	124.266	109.330	137.064	883.576	110.447	18.704
Sum	87.362	121.590	88.990	128.818	90.182	125.187	110.268	138.141	890.538		
Mean	29.121	40.530	29.663	42.939	30.061	41.729	36.756	46.047		37.106	
StdDev	41.139	56.675	41.312	59.993	41.727	58.363	51.318	64.359			52.973



Intel Trace Analyzer: пример использования

Профиль: point-to-point операции



Intel Trace Analyzer: что можно получить

1. Timeline: детальное исследование времени жизни программы
2. Статистику использования MPI функций
3. Ранк-к-ранку статистику использования сети: размеры передаваемых данных, время передачи, среднюю характеристику канала
4. Статистику использования коллективных функций MPI для каждого ранка

Документация и ссылки

1. Intel Cluster Toolkit (IMPI + IMC + ITAC) – вебсайт пакета
2. Automated MPI Correctness Checking – статья об использовании ITAC
3. Intel Cluster Toolkit Tutorial – пошаговое обучении использованию инструментов

Вопросы?


intel®
Leap ahead™