PARALLEL SOFTWARE TECHNOLOGIES LABORATORY PSTL

Performance Models for Communication in Collective I/O Operations

Shweta Jha and Edgar Gabriel

Parallel Software Technologies Lab
Department of Computer Science,
University of Houston, Houston, USA
Email: {sjha2, egabriel}@uh.edu

UNIVERSITY of HOUSTON

Agenda

- Introduction and Motivation
  - Data Decomposition Strategies
  - File Partitioning Strategies
- Performance Models
- Discussion & Evaluation
- Conclusion & Future Work

I/O demand in HPC

- Study by LLNL (2005) on I/O requirements for HPC applications:
  - 1 GB/s I/O bandwidth required per Teraflop compute capability
  - On average, 5 times more write operations than read operations
- Some Current High-End Systems

System	Peak Compute Performance	Storage Capacity	Peak Storage Bandwidth	Percentage of required I/O Bandwidth
Edison	2.2 PFLOPS	6.4 PBytes	140 GB/s	6.2%
Shaheen II	7.2 PFLOPS	17.6 PBytes	500 GB/s	6.7%
Blue Waters	13.1 PFLOPS	25 PBytes	1.18 TB/s	9.0%
Sequoia	20 PFLOPS	55 PBytes	1.5 TB/s	7.5%

Individual I/O in parallel applications

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32

Process 0:

```
read(…, offset=0,  length=2)
read(…, offset=8,  length=2)
read(…, offset=16, length=2)
read(…, offset=24, length=2)
```

- Individual Read/Write operations lead to many, small I/O requests from each process
- Arbitrary order of I/O requests from the file system perspective

⇨ suboptimal performance

Collective I/O in parallel applications

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32

Process 0:

```
read(…, offset=0,  length=4)
MPI_Send (…,length=2,dest=3,…)
read(…, offset=8,  length=4)
MPI_Send (…,length=2,dest=3,…)
read(…, offset=16, length=4)
MPI_Send (…,length=2,dest=3,…)
read(…, offset=24, length=4)
MPI_Send (…,length=2,dest=3,…)
```

- Collective I/O:
  - Fewer and larger I/O requests posted
  - Coordinates I/O access across processes

Collective I/O operations(I)

- Collective operation for reading/writing data allows to combine data of multiple processes to optimize disk-access
- Example for a collective write operation

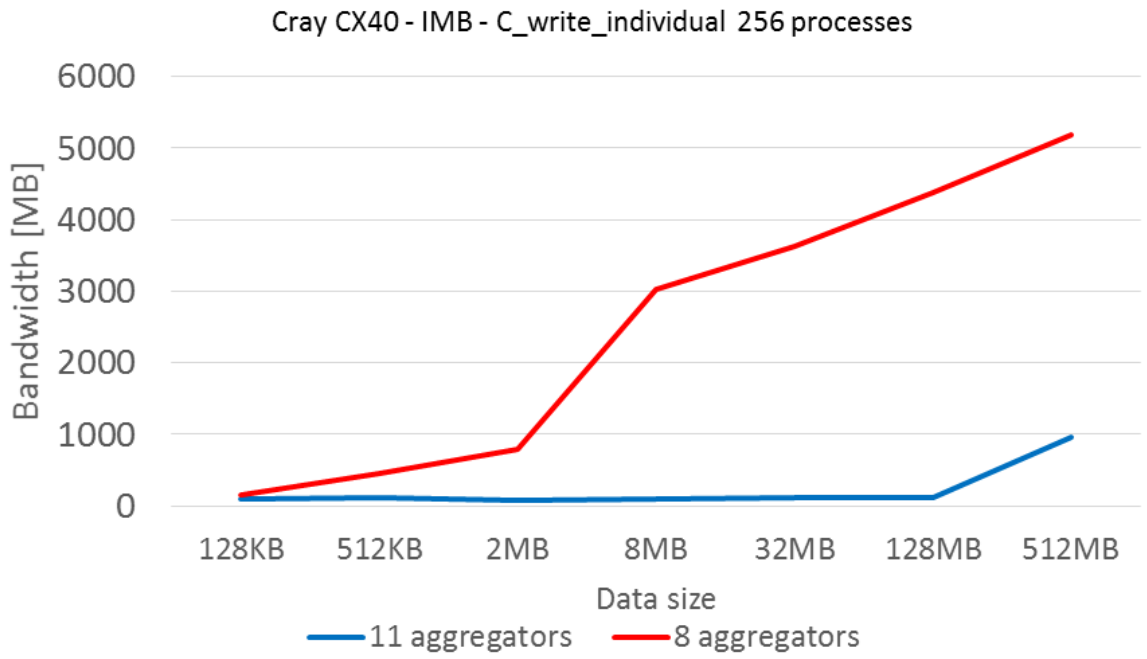
  Step 1:
  - gather data from multiple processes onto a subset of processes => aggregator processes
  - Sort data based on the offset in the file

  Step 2: aggregators write data
- Large read/write operations split the into multiple cycles internally
  - Limits the size of temporary buffers
  - Allows to overlap communication and I/O operation

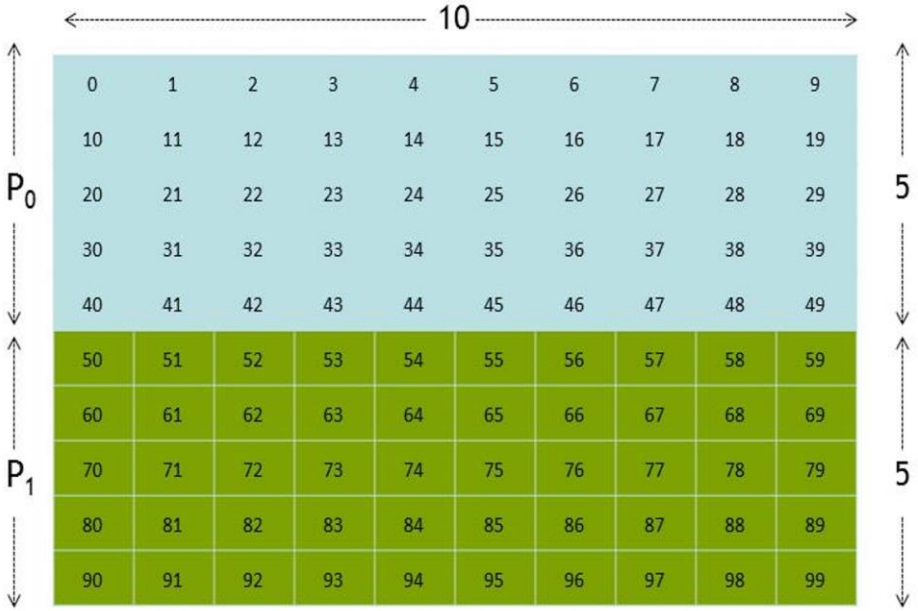
Collective I/O operations

- Collective I/O performance is highly sensitive to internal parameter values and file system characteristics

  => Analytical models for collective file I/O required

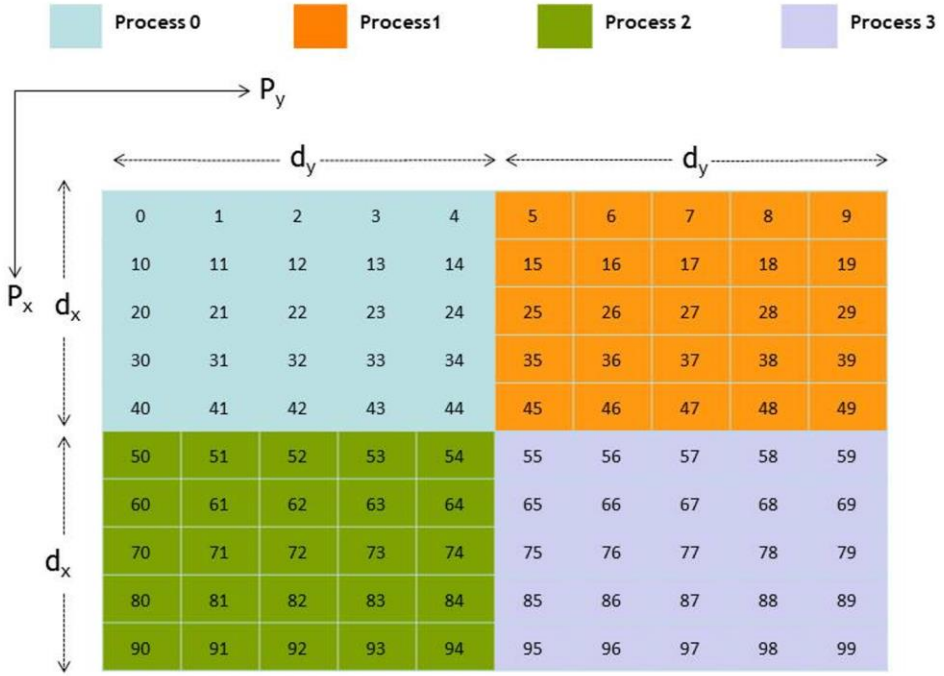


Data Decomposition

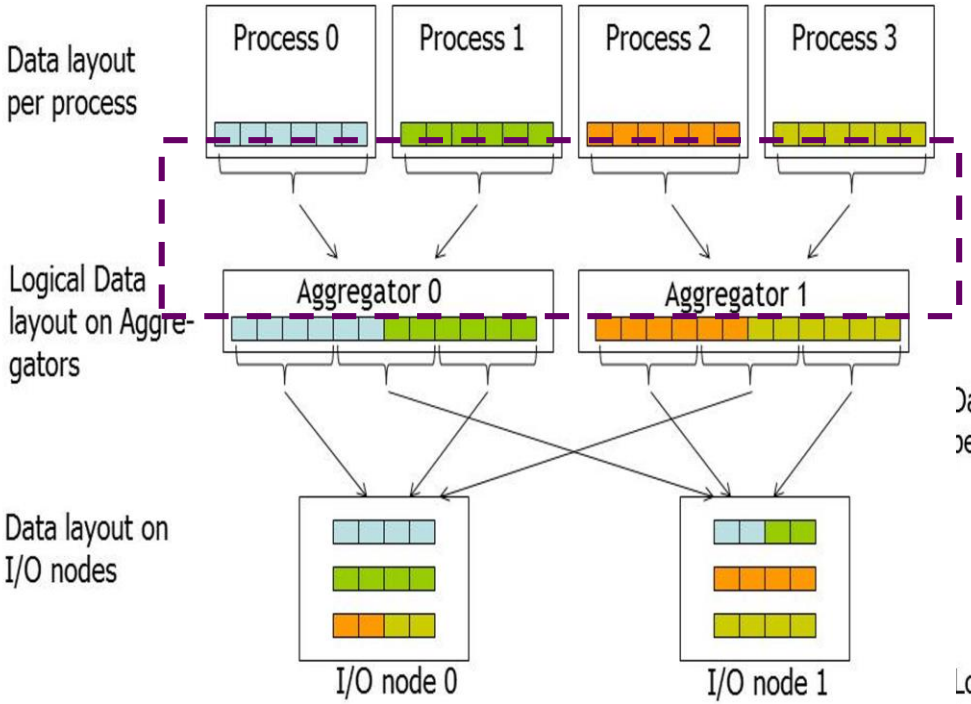


1D data decomposition of 2D matrix

2D data decomposition of 2D matrix

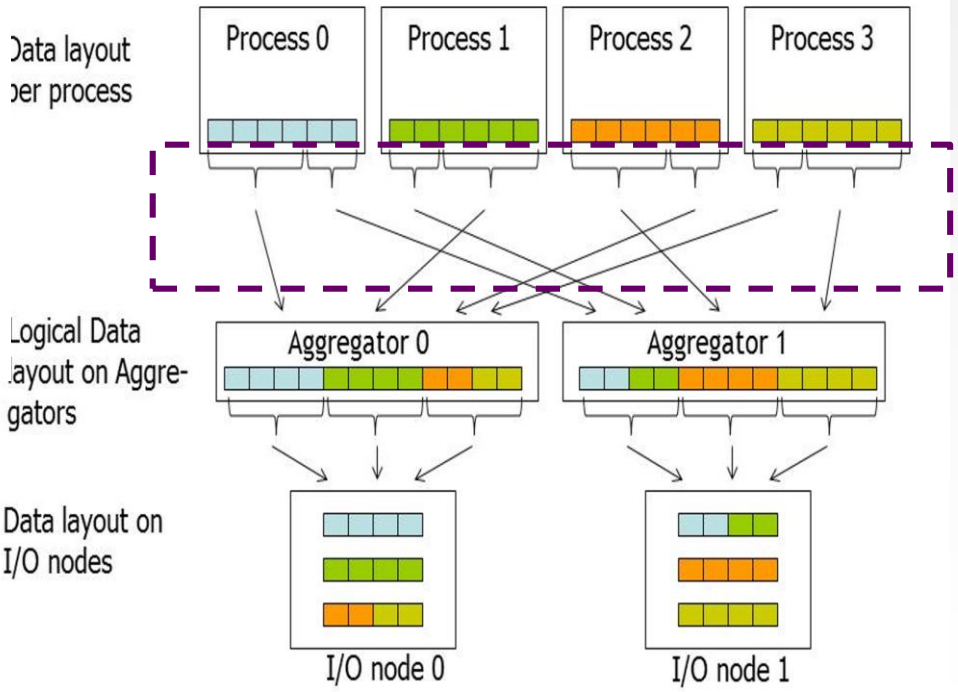


Even & Static File Partitioning



Even File Partitioning

Static File Partitioning



1-D Data Decomposition & Even File Partitioning

Data per process = 100 elements ; Collective buffer size = 50 elements

Data Distribution on Processes

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249
250	251	252	253	254	255	256	257	258	259
260	261	262	263	264	265	266	267	268	269
270	271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288	289
290	291	292	293	294	295	296	297	298	299

P_0 P_1 P_2

Data Distribution on Aggergator

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249
250	251	252	253	254	255	256	257	258	259
260	261	262	263	264	265	266	267	268	269
270	271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288	289
290	291	292	293	294	295	296	297	298	299

A_0 A_1

1 Dimensional Data & Even File Partitioning

Iteration 1:
0 to 49: From Process 0 to Aggregator 0

150 to 199: From Process 1 to Aggregator 1

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249
250	251	252	253	254	255	256	257	258	259
260	261	262	263	264	265	266	267	268	269
270	271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288	289
290	291	292	293	294	295	296	297	298	299

1 Dimensional Data & Even File Partitioning

Iteration 2:

50 to 99: From Process 0 to Aggregator 0

200 to 249: From Process 2 to Aggregator 1

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249
250	251	252	253	254	255	256	257	258	259
260	261	262	263	264	265	266	267	268	269
270	271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288	289
290	291	292	293	294	295	296	297	298	299

1 Dimensional Data & Even File Partitioning

Iteration 3:

100 to 149: From Process 1 to Aggregator 0

250 to 299: From Process 2 to Aggregator 1

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249
250	251	252	253	254	255	256	257	258	259
260	261	262	263	264	265	266	267	268	269
270	271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288	289
290	291	292	293	294	295	296	297	298	299

1 Dimensional Data & Static File Partitioning

Data per process = 100 elements ; Collective buffer size = 50 elements

Data Distribution on Processes

P_0	0	1	2	3	4	5	6	7	8	9
	10	11	12	13	14	15	16	17	18	19
	20	21	22	23	24	25	26	27	28	29
	30	31	32	33	34	35	36	37	38	39
	40	41	42	43	44	45	46	47	48	49
	50	51	52	53	54	55	56	57	58	59
	60	61	62	63	64	65	66	67	68	69
	70	71	72	73	74	75	76	77	78	79
	80	81	82	83	84	85	86	87	88	89
	90	91	92	93	94	95	96	97	98	99
P_1	100	101	102	103	104	105	106	107	108	109
	110	111	112	113	114	115	116	117	118	119
	120	121	122	123	124	125	126	127	128	129
	130	131	132	133	134	135	136	137	138	139
	140	141	142	143	144	145	146	147	148	149
	150	151	152	153	154	155	156	157	158	159
	160	161	162	163	164	165	166	167	168	169
	170	171	172	173	174	175	176	177	178	179
	180	181	182	183	184	185	186	187	188	189
	190	191	192	193	194	195	196	197	198	199
P_2	200	201	202	203	204	205	206	207	208	209
	210	211	212	213	214	215	216	217	218	219
	220	221	222	223	224	225	226	227	228	229
	230	231	232	233	234	235	236	237	238	239
	240	241	242	243	244	245	246	247	248	249
	250	251	252	253	254	255	256	257	258	259
	260	261	262	263	264	265	266	267	268	269
	270	271	272	273	274	275	276	277	278	279
	280	281	282	283	284	285	286	287	288	289
	290	291	292	293	294	295	296	297	298	299

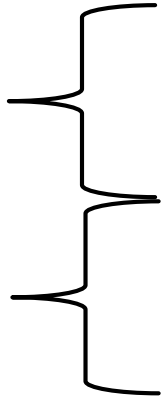
Data Distribution on Aggergator

A_0	0	1	2	3	4	5	6	7	8	9
	10	11	12	13	14	15	16	17	18	19
	20	21	22	23	24	25	26	27	28	29
	30	31	32	33	34	35	36	37	38	39
	40	41	42	43	44	45	46	47	48	49
A_1	50	51	52	53	54	55	56	57	58	59
	60	61	62	63	64	65	66	67	68	69
	70	71	72	73	74	75	76	77	78	79
	80	81	82	83	84	85	86	87	88	89
	90	91	92	93	94	95	96	97	98	99
A_0	100	101	102	103	104	105	106	107	108	109
	110	111	112	113	114	115	116	117	118	119
	120	121	122	123	124	125	126	127	128	129
	130	131	132	133	134	135	136	137	138	139
	140	141	142	143	144	145	146	147	148	149
A_1	150	151	152	153	154	155	156	157	158	159
	160	161	162	163	164	165	166	167	168	169
	170	171	172	173	174	175	176	177	178	179
	180	181	182	183	184	185	186	187	188	189
	190	191	192	193	194	195	196	197	198	199
A_0	200	201	202	203	204	205	206	207	208	209
	210	211	212	213	214	215	216	217	218	219
	220	221	222	223	224	225	226	227	228	229
	230	231	232	233	234	235	236	237	238	239
	240	241	242	243	244	245	246	247	248	249
A_1	250	251	252	253	254	255	256	257	258	259
	260	261	262	263	264	265	266	267	268	269
	270	271	272	273	274	275	276	277	278	279
	280	281	282	283	284	285	286	287	288	289
	290	291	292	293	294	295	296	297	298	299

1 Dimensional Data & Static File Partitioning

Iteration 1:
0 to 49: From Process 0 to Aggregator 0

50 to 99: From Process 0 to Aggregator 1



0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249
250	251	252	253	254	255	256	257	258	259
260	261	262	263	264	265	266	267	268	269
270	271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288	289
290	291	292	293	294	295	296	297	298	299

1 Dimensional Data & Static File Partitioning

Iteration 2:

100 to 149: From Process 1 to Aggregator 0

150 to 199: From Process 1 to Aggregator 1

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249
250	251	252	253	254	255	256	257	258	259
260	261	262	263	264	265	266	267	268	269
270	271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288	289
290	291	292	293	294	295	296	297	298	299

1 Dimensional Data & Static File Partitioning

Iteration 3:

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249
250	251	252	253	254	255	256	257	258	259
260	261	262	263	264	265	266	267	268	269
270	271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288	289
290	291	292	293	294	295	296	297	298	299

200 to 249: From Process 2 to Aggregator 0

250 to 299: From Process 2 to Aggregator 1

Generic model (I)

Starting point: LogGP performance model

Assumptions:

- Equal communication costs between any pair of processes
- Equal Data per Processor d_p

Number of Processes P

Number of Aggregator P_a

Collective Buffer size b_c

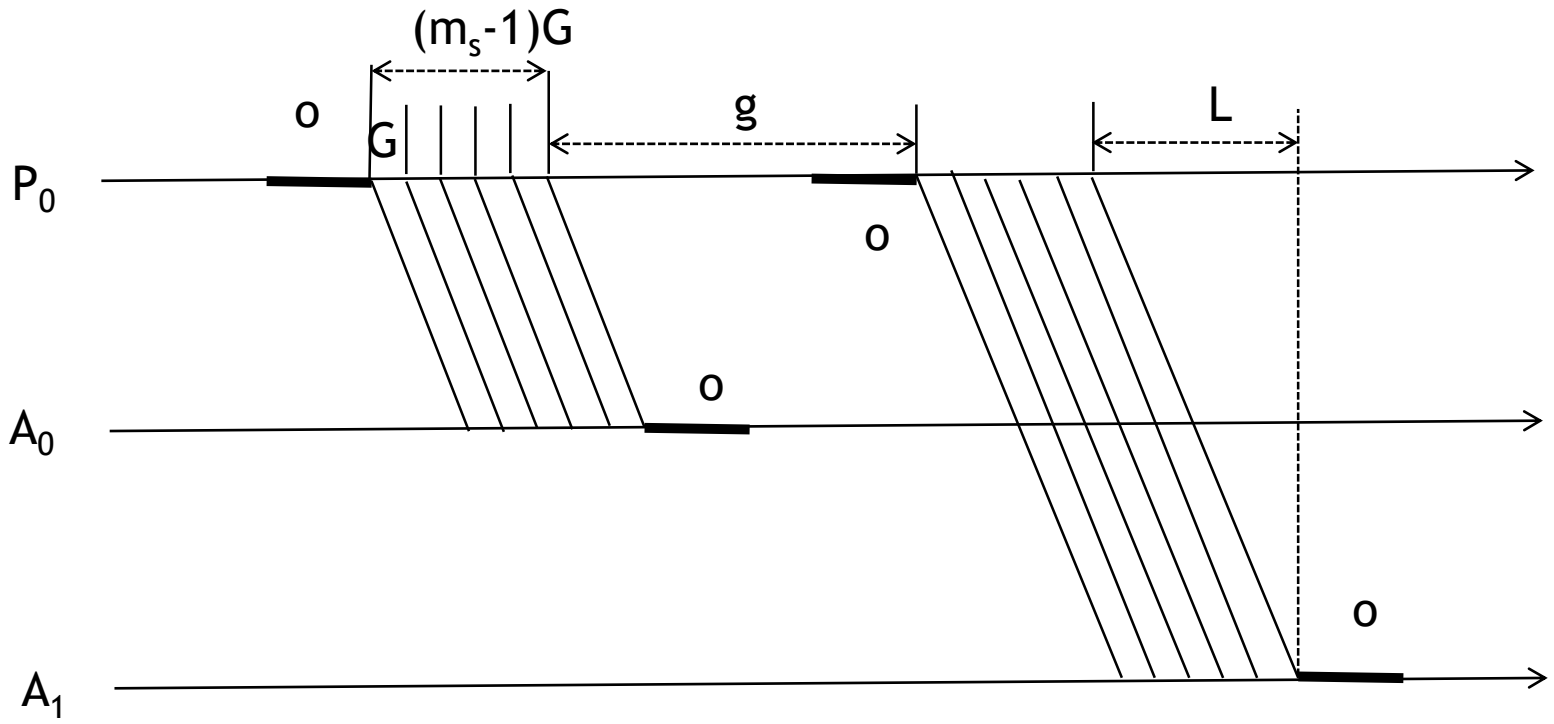
Aggregator's file domain b_a

Aggregator's file domain $b_a = (P \cdot d_p) / P_a$

As aggregators are also MPI processes, send & receive do not overlap

$$T = T_{send} + T_{recv}$$

Generic model (II)



$T_{send} = n_s(L+2o+(n_{as}-1)g + n_{as}(m_s-1)G)$ for n_s send cycles

$T_{recv} = n_r(L+2o+(n_{ar}-1)g + n_{ar}(m_s-1)G)$ for n_r receive cycles

n_{as} Number of aggregators each processes communicate with

n_{ar} Number of processes each aggregator receives from

Generic model (III)

- n_s (number of send cycles) = $\lceil d_p / (n_{as} * m_s) \rceil$
- n_r (number of receive cycles) = $\lceil b_a / b_c \rceil$

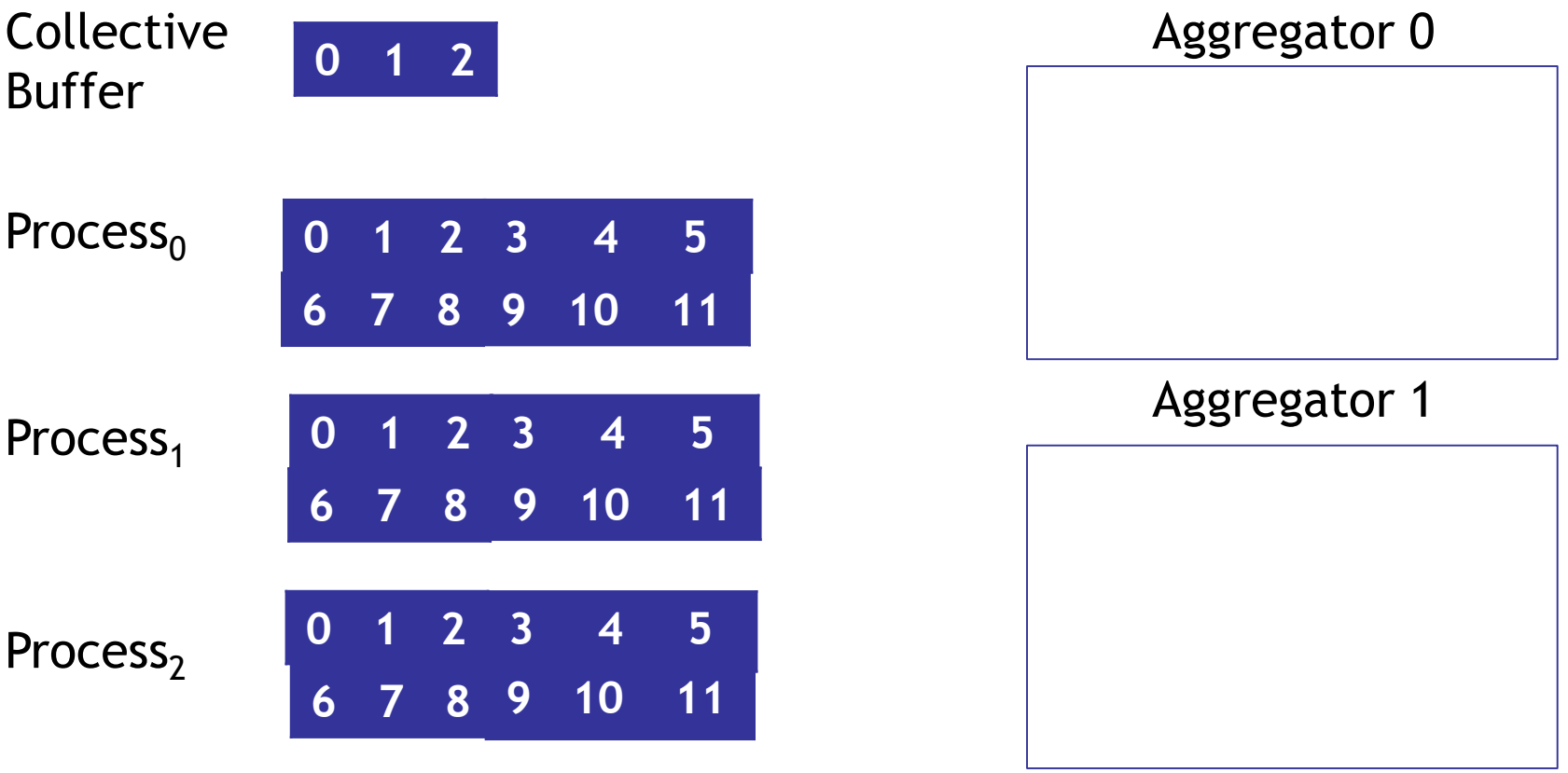
  = $\lceil (P * d_p) / (P_a * b_c) \rceil$
- Terms to be determined:

  n_{as} , n_{ar} , m_s
- n_{as} , n_{ar} , m_s values can be derived in terms of d_p, P, P_a, b_c etc

	Even-1D		Even-2D				Static-1D	Static-2D		
	$d_p > b_c$	$d_p \leq b_c$	$P_a = P_x$	$P_a = c \cdot P_x$	$P_a > P_x$	$P = \frac{P_x}{c}$		$d_y \cdot P_y = b_c$	$d_y \cdot P_y = c \cdot b_c$	$c' \cdot d_y \cdot P_y = b_c$
n_{as}	1	1	1	c	$\lceil c \rceil$ where $c = \frac{P_x}{P_a}$	$\lceil \frac{P_x}{P_a} \rceil$	P_a if $d_p > P_a \cdot b_c$; $\lceil \frac{d_p}{P_a} \rceil$ else	$P_a if d_x < P_a$; d_x else	$\frac{P_a}{c}$	$P_a if d_x \geq c' \cdot P_a$; $\frac{d_x}{c'}$ else
n_{ar}	1	$\lceil \frac{b_c}{d_p} \rceil$	P_y	P_y	P_y	P_y	1	P_y	$\frac{P_y}{c}$	P_y
m_s	b_c	d_p	P_a if $d_x \geq c' \cdot P_a$; $\frac{d_x}{c'}$ else	P_a if $d_x \geq c' \cdot P_a$; $\frac{d_x}{c'}$ else	P_a if $d_x \geq c' \cdot P_a$; $\frac{d_x}{c'}$ else	P_a if $d_x \geq c' \cdot P_a$; $\frac{d_x}{c'}$ else	b_c	d_y	d_y	$c' \cdot d_y$

Example: Even 1-D

Data per process d_p=12 > Collective Buffer Size b_c= 3



Message length = Collective Buffer
$n_{ar} = n_{as} = 1$

send iteration = $d_p/b_c = 12/3$
recv iteration = $b_a/b_c = 18/3$

A case study

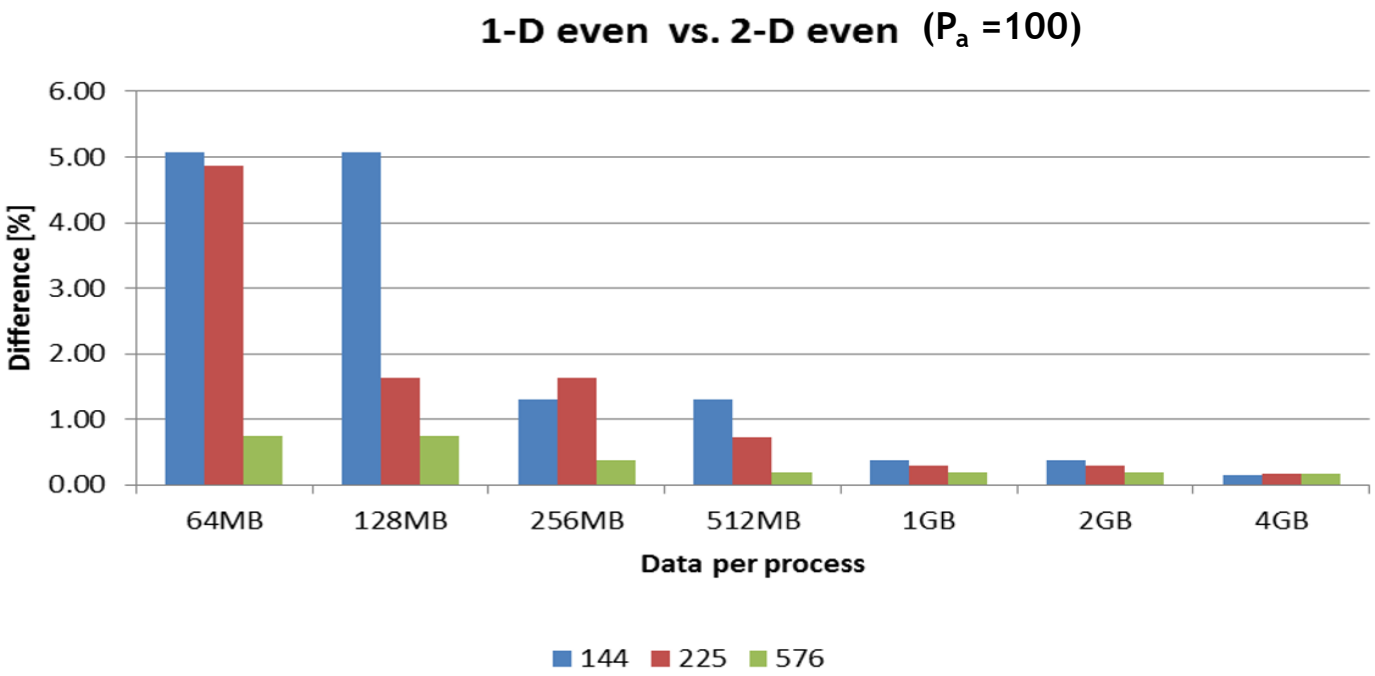
Parameter	even 1D			even 2D			Static 1D			Static 2D		
# of Process	144	225	576	144	225	576	144	225	576	144	225	576
nas	1	1	1	6	5	3	32	32	32	64	64	64
ns	32	32	32	64	96	256	1	1	1	6	8	12
nar	1	1	1	12	15	24	1	1	1	12	15	24
nr	72	113	288	72	113	288	72	113	288	72	113	288
ms	32MB	32MB	32MB	2.66MB	2.13MB	1.33MB	32MB	32MB	32MB	2.66MB	2.13MB	1.33MB

data size = 1GB/process, collective buffer size =32MB & #aggregator = 64

Observations:

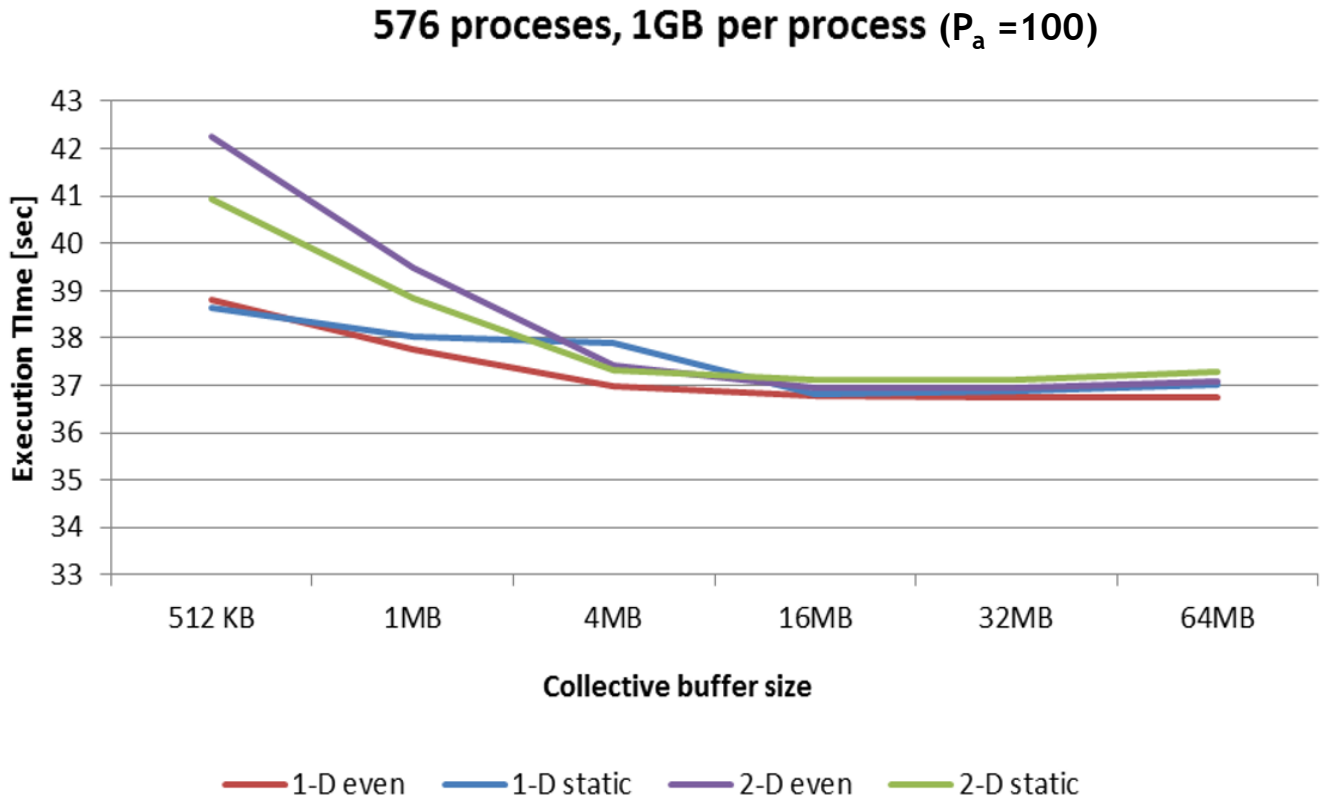
- Sender attributes (n_{as}, n_s) are case dependent, but receiver attributes (n_{ar}, n_r) are similar in even and static
- 2D distribution uses smaller m_s but larger no. of messages
- Even file part. : 32 cycles with a single message of length b_c sent
  Static file part.: 1 Cycle with 32 messages of length b_c sent

Discussion (I): static vs. even file partitioning



- 2-D data distributions has up to 5% higher communication costs compared to the 1-D data distribution
- The difference between the two data distributions decreases with increase in data per process

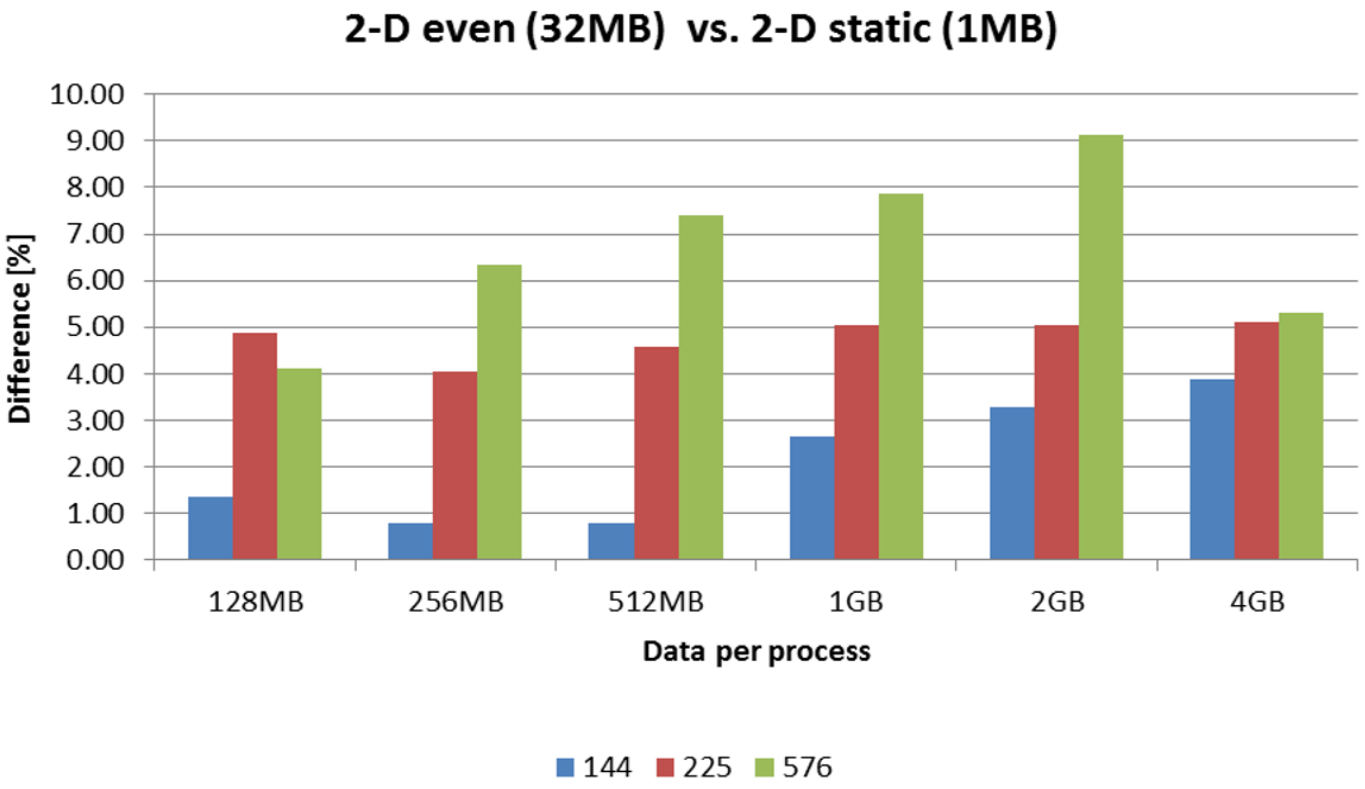
Discussion (III)



Influence of collective buffer size

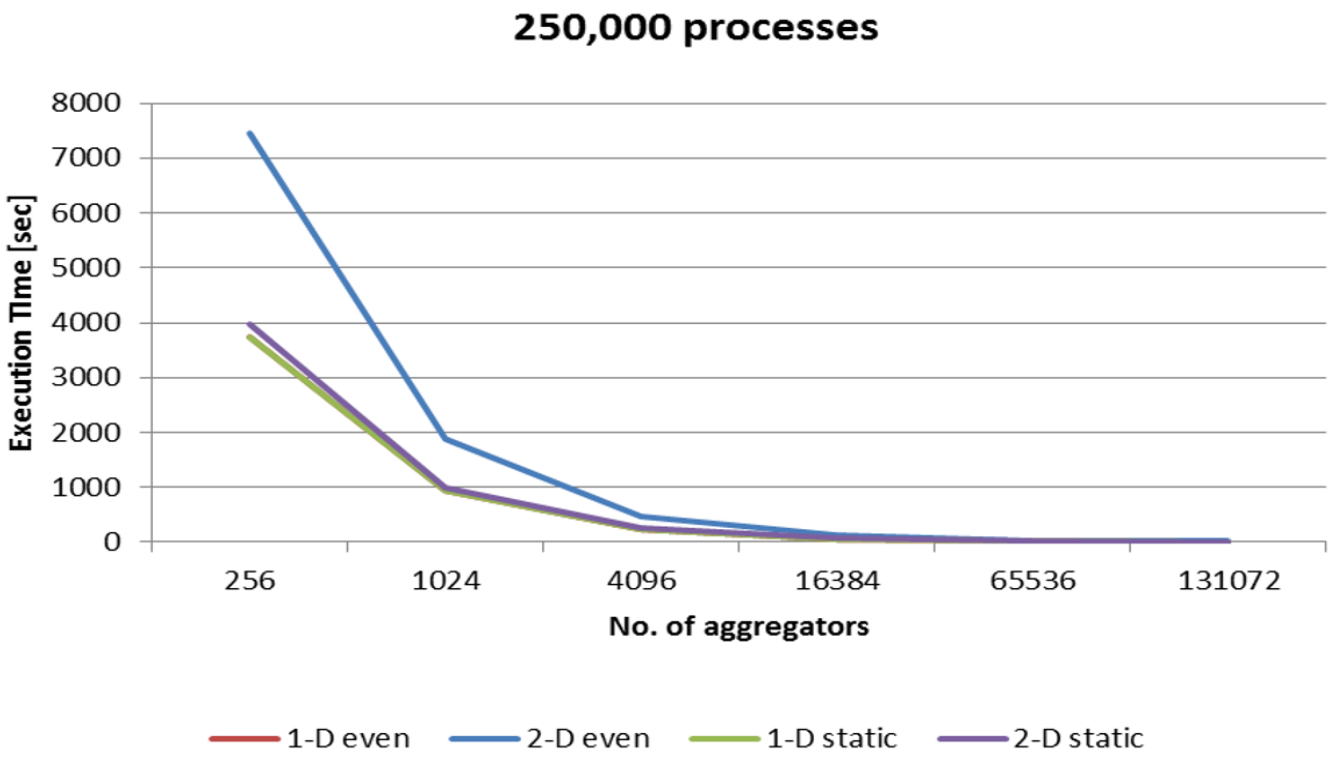
- Performance improvements while increasing the collective buffer size from lower values up to 16MB

Discussion (IV)



- static file partitioning with a collective buffer size of 1MB would lead to a significant performance degradation compared to the even file partitioning with 32 MB collective buffer size

Discussion (V)



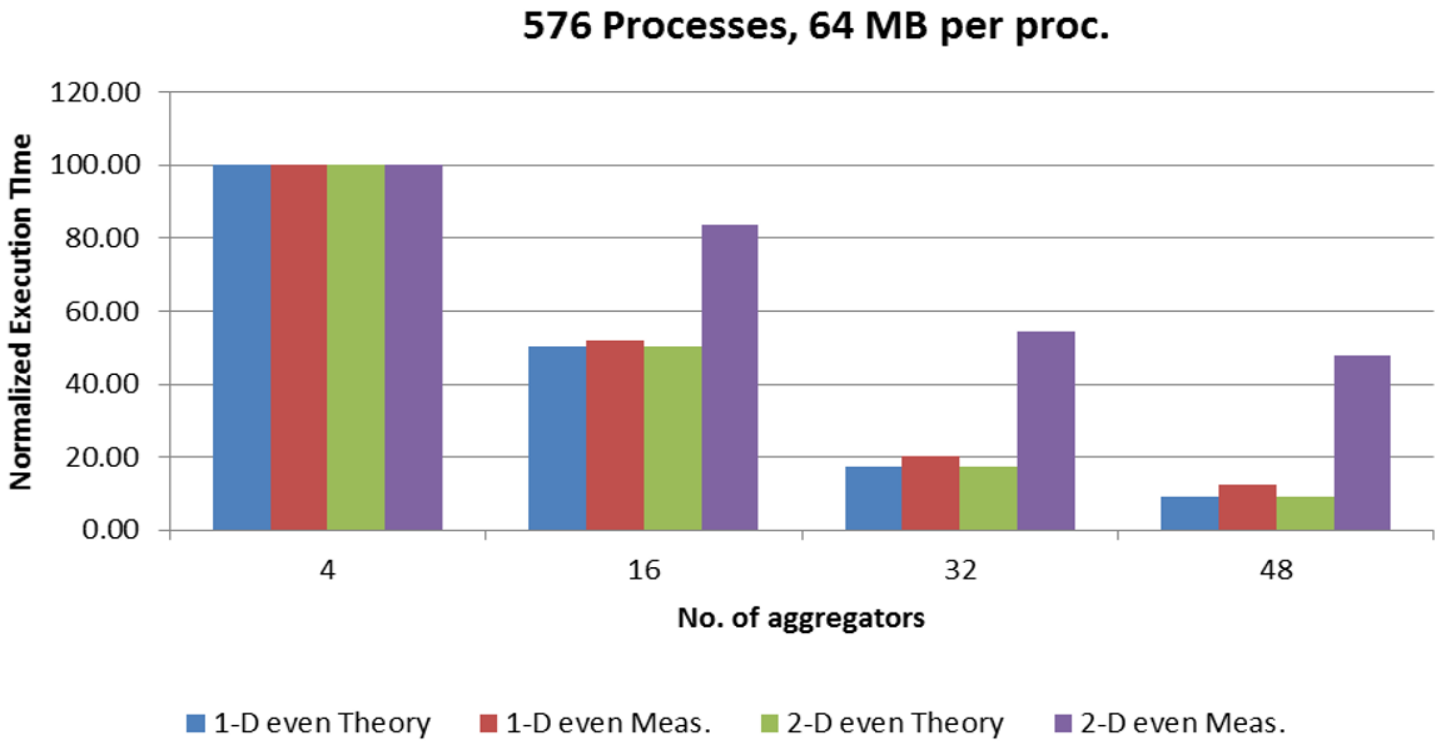
Projection for large process counts

- 8k - 16k aggregators to minimize the communication costs of the collective write operation.

Evaluation

- Platforms:
  - Opuntia (University of Houston) - 56Gbit Ethernet network
  - Crill cluster (crill-IB) (University of Houston) - DDR InfiniBand network
  - Crill cluster (crill-GE) (University of Houston) - Gigabit Ethernet network
- Process counts: 36, 64, 144, 256, 576
- Data sizes:
  - (3,800 * 3,800) bytes (13MB) per process
  - (8, 192 * 8, 192) bytes (64 MB) per process
  - (16, 384 * 16, 384) bytes (256 MB) per process
  - file sizes 468MB to 144GB
- LogGP parameters determined using ‘Netgauge 2.4.6’

Evaluation (II)



- Performance model predicts the measured performance reasonably well

Evaluation (III)

- Case 1: varying the datasize per process (process count and number of aggregators are constant)

	match	mismatch
even 1D	37	0
even 2D	36	0
static 1D	40	0
static 2D	20	0

- Case 2: varying the number of aggregators (process count and the datasize are constant)

	match	mismatch
even 1D	38	4
even 2D	39	3
static 1D	37	8
static 2D	36	7

- Case 3: varying process count (the number of aggregators and the datasize are constant)

	match	mismatch
even 1D	17	3
even 2D	17	2
static 1D	20	1
static 2D	19	1

*Match: Increase in variable predicts the correct trend in communication cost
*Mismatch: Increase in variable predicts the incorrect trend in communication cost

Summary and Future Work

- Developed performance models for the communication occurring in collective I/O operations taking data decomposition and file partitioning strategy into account
- A good match between predicted and observed behavior
  - Some discrepancies due to limitations of the LogGP model and simplifications introduced by our models

- Future work:
  - Extending the models to collective read operations
  - Including actual I/O operations into the models
    - Very challenging: models have to capture locking protocol and caching effects used by file system