Lv.的博客

Avatarx

Qt 格式化字符串

Qt字符串格式化性能比较

Qt字符串格式方法有三种, QString::arg(), QString::sprinft()和QStringList::join().
今天我做了个简单的性能测试, 希望对各位有所帮助.

调用QString::arg()一次:
复制代码

```
1. QString s("1234567890");

    // 开始计时
    for (int i = 0; i < 10000; ++i) {
        QString str("%1");
        str.arg(s);
    }
    // 停止计时
```

调用QString::arg()十次:
复制代码

```
1. QString s("1234567890");

    // 开始计时
    for (int i = 0; i < 10000; ++i) {
        QString str("%1%2%3%4%5%6%7%8%9%10");
        str
            .arg(s)
            .arg(s)
            .arg(s)
            .arg(s)
            .arg(s)
            .arg(s)
            .arg(s)
            .arg(s)
            .arg(s)
            .arg(s);
    }
    // 停止计时
```

调用QString::sprinft()一次:
复制代码

```
1. char s2[] = {"1234567890"};

        // 开始计时
        for (int i = 0; i < times; ++i) {
        QString().sprintf("%d", s2);
        }
        // 停止计时
```

昵称: Avatarx
园龄: 5年10个月
粉丝: 198
关注: 10
+加关注

< 2019年12月 >

日	一	二	三	四	五	六
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	1	2	3	4
5	6	7	8	9	10	11

搜索

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

qt mvc(4)
qt pro(2)
qt python(1)
qt QSettings(1)
qt qt5 基础(1)
qt sql 数据库操作(1)
qt stylesheet 美化 控件样式(1)
qt 编译oracle驱动 oci(1)
qt 程序插件(1)
qt 程序居中显示(1)
更多

随笔分类

线程的同步与异步
.Net Core(3)
ACE(2)
AI(人工智能)(73)
Android
ARM(28)
boost库(30)
C#(154)
c++(88)
C++11(34)
codeblocks(2)
cowboy
CppCheck(1)

调用QString::sprinft()十次:
复制代码
```
1. char s2[] = {"1234567890"};


        // 开始计时
        for (int i = 0; i < times; ++i) {
        QString().sprintf("%d%d%d%d%d%d%d%d%d%d", s2, s2, s2, s2, s2, s2, s2, s2, s2,
        }
        // 停止计时
```

调用QStringList::join()一次:
复制代码
```
1. QString s("1234567890");


        // 开始计时
        for (int i = 0; i < times; ++i) {
        QStringList strList;
        strList.append(s);
        strList.join(QString());
        }
        // 停止计时
```

调用QStringList::join()十次:
复制代码
```
1. QString s("1234567890");


        // 开始计时
        for (int i = 0; i < times; ++i) {
        QStringList strList;
        strList.append(s);
        strList.append(s);
        strList.append(s);
        strList.append(s);
        strList.append(s);
        strList.append(s);
        strList.append(s);
        strList.append(s);
        strList.append(s);
        strList.append(s);
        strList.join(QString());
        }
        // 停止计时
```

测试结果:
运行一次的耗时: QString::arg() 0.412纳秒 < QStringList::join() 0.625纳秒 < QString::sprinft()
运行十次的耗时: QStringList::join() 2.426纳秒 < QString::arg() 5.175纳秒 < QString::sprinft()

现在让我们来看看这些函数是怎么完成各自的任务的.

先来看看QString::arg(). 这个函数的实现非常简单, 它先将传入的参数转换成QString格式的字符串参数, 换源字符串里的替换符(比如%1, %2, %3什么的). replaceArgEscapes首先计算一下新的字符串将会有多 再将源字符串和字符串参数合并到临时字符串中. 一次操作就这么完成了.

在这个过程中, 比较消耗时间的是创建了一个临时字符串. 随着累加调用QString::arg(), 当调用十次的时候 秒是消耗在创建10个临时字符串当中. 还有重复多次的内存拷贝. 所以QString::arg()累加调用的越多, 它的执

下面再来看看QString::sprintf(), 这个函数是仿照C函数的sprintf(), 使用方法上一致, 这个函数也不复杂, QString::vsprintf()中先创建个临时字符串, 然后类似于replaceArgEscapes的方式循环追加到这个临时字符串的大小, 调用的是QString::append(). 这个函数在缓存够大的时候直接追加数据, 如果不够大则重新分配足 QString::arg()本质上并没有太大区别. 因为每次都会生成个新的字符串, 并将数据拷贝进去.

好的东西都要放到最后面再说, 现在我们来看看QStringList::join()这个方法, 这个方法为什么合并10个数据 朗了. 它先循环获取每个字符串的长度, 这样就可以计算出整个字符串的长度, 注意, 是整个, 不是一部分. 实际上 效率永远都是O(1).

那么说到这里, QString::arg()和QString::sprintf()的方法基本一样, 为什么差距那么大呢? 嘿嘿, 你们有没有 QString(), 而给QString::sprintf传递的是char *, 所以arg执行的时候不需要再把参数转换成字符串参数了, QString::sprintf不得不将char*类型的字符串转换成为QString(). 这也是没办法的事儿, 因为QString::spri

现在我把QString::arg()和QString::join()里面预先构建的QString()字符串都替换成临时构建. 让我们来看看

运行一次的耗时: QString::arg() 0.642纳秒 < QStringList::join() 0.792纳秒 < QString::sprinft()
运行十次的耗时: QStringList::join() 4.363纳秒 < QString::arg() 7.171纳秒 < QString::sprinft()

这次算是公平的了, 从上面的数据可以看出来, 为什么官方不推荐使用QString::sprintf()这个函数了, 首先是这 C/C++程序员的使用习惯. 基本上可以说一无是处了. 那么为啥也没有推荐使用QStringList.join()呢? 主要是 father %1 has two bros, %2 was killed by malaria, %3 is still alive.", 如果用QString::arg()是如此 成这样: "My father " << "Old John" << " has two bros, " << "one" << " was killed by malaria, " 替换三个, 一个是合并7个.

我希望能通过这次测试各位从事Qt的朋友们, QString::sprintf()完全可以放弃了, 认为这个性能会如何如何好 函数; 剩下99%的时候都应该使用QString::arg() , 因为各位不会一下子格式化十个八个参数的, 一般都是一两 个则要果断选择QStringList::join()了.

写了这么多, 希望能对各位有所帮助.

补充:
非常谢谢dbzhang800" data-card-url="pw_ajax.php?action=smallcard&type=showcard&uid=" tarc checkUrl(this)" id="url_1">dbzhang800的补充, QString::arg()的模板函数版本平时真的没怎么用过, 哈 >=10的参数, 所以把十个参数改成九个参数进行比较.
下面是最新的性能比较结果.
运行
运行一次的耗时: QStringBuilder 0.362纳秒 < QString::arg() 0.636纳秒 < QString::arg() 模板版本 纳秒 < QString::sprinft() 1.275纳秒;
运行十次的耗时: QStringBuilder 2.126纳秒 < QStringList::join() 3.781纳秒 < QString::arg() 模板 纳秒 < QString::sprinft() 8.933纳秒;

从上面的运行结果来看, QString::arg()模板版本只是略微强于QString::arg()的多次调用版本. 函数内部仍然 式仍然面临多次分配内存的操作. 而且QString::arg()模板版本其实并不是模板函数, 只是一次接受多个QStrin 它已经丢失了QString::arg()普通版本的一大优势--基本类型通杀. 也就是说, 如果你要使用QString::arg()的 据都自己转换成QString类型的, 否则你就没办法使用.
牛叉的总要放在后面说, 是吧. dbzhang800" data-card-url="pw_ajax.php?action=smallcard&type= onclick="return checkUrl(this)" id="url_2">dbzhang800翻译的文章中已经对这个有详细的介绍了. 使用 个是内建的, 然后注意字符串连接时候的连接符是%, 而不是+.

再次感谢dbzhang800" data-card-url="pw_ajax.php?action=smallcard&type=showcard&uid=" tarc checkUrl(this)" id="url_4">dbzhang800的帮助, 希望大家能够继续补充!

dbzhang800

呵呵，写的真不短啊。我补充一点。

都是QString的话，如果用 arg的话，应该选择:
QString("%1 %2 %3 %4").arg(s1, s2, s3, s4)
而不是
QString("%1 %2 %3 %4").arg(s1).arg(s2).arg(s3).arg(s4)

随笔档案

另外字符串链接的话：
s1 + s2 + s3 + s4
也是比较常用的，但性能就不如
s1 % s2 % s3 % s4
这种写法了

分类: qt

Avatarx
关注 - 10
粉丝 - 198
+加关注

0 0

« 上一篇： 实现C++模板类头文件和实现文件分离的方法
» 下一篇： 搭建Windows下Java Web开发环境

posted @ 2015-02-16 10:20 Avatarx 阅读(15689) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

(评论功能已被禁用)

相关博文:
· 趣味算法：字符串反转的N种方法
· QT 格式化字符串功能
· Java字符串之性能优化
· Qtsprintf_s函数格式化字符串出错
· Java字符串之性能优化
» 更多推荐...

最新 IT 新闻:
· 李洪元回应华为声明：大家看看先，我听全国人民的
· 逆康普顿散射打造"史上最强的光"
· 暴风集团：高管已全部离职，公司仅剩10余人
· 引力波天文学开启黑洞研究新纪元
· 金山云拟发行约5509万股D+系列优先股
» 更多新闻...

2017年6月(12)
2017年5月(41)
2017年4月(36)
2017年3月(56)
2017年2月(18)
2017年1月(10)
2016年12月(8)
2016年11月(5)
2016年10月(2)
2016年9月(1)
2016年8月(1)
2016年7月(14)
2016年6月(71)
2016年5月(63)
2016年4月(76)
2016年3月(43)
2016年2月(1)
2016年1月(9)
2015年12月(1)
2015年11月(31)
2015年10月(30)
2015年9月(87)
2015年8月(10)
2015年7月(12)
2015年6月(7)
2015年5月(97)
2015年4月(52)
2015年3月(53)
2015年2月(7)
2015年1月(49)
2014年12月(61)
2014年11月(58)
2014年10月(34)
2014年9月(5)
2014年8月(15)
2014年7月(41)
2014年6月(68)
2014年5月(144)
2014年4月(4)

最新评论

1. Re:LabView培训
链接没有了楼主
--梦游城市

2. Re:Qt 类外调用一个 private slots 函数
鬼斧神工的做法，见识了。谢谢。谢谢你那么多有水平的文章。
--tadpole999

3. Re:qt QThread
你好，我最近也在弄qt线程，但是会发现一个问题，就是正常运行都没问题，但是如果线程执行过程中，用户强制退出程序的时候，此时进行析构的时候会出错，这怎么解决啊?
--yzfree

阅读排行榜

1. 详细的OS X Yosemite 10.10懒人版安装教程(29723)
2. wireshark如何抓取本机包(29554)
3. c# sleep 例子(25570)
4. qt超强精美绘图控件 - QCustomPlot一览 及 安装使用教程(22979)
5. 5 个免费的受欢迎的 SQLite 管理工具(21341)

评论排行榜

1. qt QThread(1)
2. Qt 类外调用一个 private slots 函数(1)
3. LabView培训(1)

推荐排行榜

1. wireshark如何抓取本机包(7)
2. C# volatile 关键字(4)
3. EF中使用SQL语句或存储过程(2)
4. Linux驱动开发学习的一些必要步骤(2)
5. udp 不需要 listen(2)